

IC Compiler

Design Planning

User Guide

Version C-2009.06, June 2009

SYNOPSYS®

Copyright Notice and Proprietary Information

Copyright © 2009 Synopsys, Inc. All rights reserved. This software and documentation contain confidential and proprietary information that is the property of Synopsys, Inc. The software and documentation are furnished under a license agreement and may be used or copied only in accordance with the terms of the license agreement. No part of the software and documentation may be reproduced, transmitted, or translated, in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without prior written permission of Synopsys, Inc., or as expressly provided by the license agreement.

Right to Copy Documentation

The license agreement with Synopsys permits licensee to make copies of the documentation for its internal use only. Each copy shall include all copyrights, trademarks, service marks, and proprietary rights notices, if any. Licensee must assign sequential numbers to all copies. These copies shall contain the following legend on the cover page:

"This document is duplicated with the permission of Synopsys, Inc., for the exclusive use of _____ and its employees. This is copy number _____. "

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Registered Trademarks (®)

Synopsys, AMPS, Astro, Behavior Extracting Synthesis Technology, Cadabra, CATS, Certify, CHIPit, Design Compiler, DesignWare, Formality, HDL Analyst, HSIM, HSPICE, Identify, iN-Phase, Leda, MAST, ModelTools, NanoSim, OpenVera, PathMill, Physical Compiler, PrimeTime, SCOPE, Simply Better Results, SiVL, SNUG, SolvNet, Syndicated, Synplicity, Synplify, Synplify Pro, Synthesis Constraints Optimization Environment, TetraMAX, the Synplicity logo, UMRBus, VCS, Vera, and YIELDirector are registered trademarks of Synopsys, Inc.

Trademarks (™)

AFGen, Apollo, Astro-Rail, Astro-Xtalk, Aurora, AvanWaves, BEST, Columbia, Columbia-CE, Confirma, Cosmos, CosmosLE, CosmosScope, CRITIC, CustomSim, DC Expert, DC Professional, DC Ultra, Design Analyzer, Design Vision, DesignerHDL, DesignPower, DFTMAX, Direct Silicon Access, Discovery, Eclypse, Encore, EPIC, Galaxy, Galaxy Custom Designer, HANEX, HAPS, HapsTrak, HDL Compiler, Hercules, Hierarchical Optimization Technology, High-performance plus ASIC Prototyping System, HSIM, i-Virtual Stepper, IIICE, in-Sync, iN-Tandem, Jupiter, Jupiter-DP, JupiterXT, JupiterXT-ASIC, Liberty, Libra-Passport, Library Compiler, Magellan, Mars, Mars-Rail, Mars-Xtalk, Milkyway, ModelSource, Module Compiler, MultiPoint, Physical Analyst, Planet, Planet-PL, Polaris, Power Compiler, Raphael, Saturn, Scirocco, Scirocco-i, Star-RCXT, Star-SimXT, System Compiler, System Designer, Taurus, TotalRecall, TSUPREM-4, VCS Express, VCSI, VHDL Compiler, VirSim, and VMC are trademarks of Synopsys, Inc.

Service Marks (SM)

MAP-in, SVP Café, and TAP-in are service marks of Synopsys, Inc.

SystemC is a trademark of the Open SystemC Initiative and is used under license.

ARM and AMBA are registered trademarks of ARM Limited.

Saber is a registered trademark of SabreMark Limited Partnership and is used under license.

All other product or company names may be trademarks of their respective owners.

Contents

What's New in This Release	xviii
About This Guide	xviii
Customer Support	xxi
1. Introduction to Design Planning	
Overview	1-2
Why Plan the Design?	1-3
Using a Hierarchical Methodology	1-4
Hierarchical Methodology for Design Planning	1-4
Design Partitioning	1-6
Hierarchical Floorplanning	1-8
Hierarchical Timing Closure	1-11
2. Initializing the Floorplan	
Reading the I/O Constraints	2-2
Loading the Top-Level I/O Pad and Pin Constraints	2-2
Setting Physical Constraints on I/O Pads	2-3
Setting Chip-Level Constraints on I/O Pads	2-4
Reporting the Pin and Pad Physical Constraints	2-4
Removing the Pin and Pad Physical Constraints	2-5
Working With Multiheight I/Os	2-5
Adjusting the Spacing and Location of I/O Cells	2-6
Defining the Core and Placing the I/O Pads	2-7

Creating Rectilinear Core Areas	2-10
Writing I/O Constraint Information	2-14
Reading Synopsys Design Constraints	2-15
Adding Cell Rows	2-15
Removing Cell Rows	2-16
Managing Tasks in Parallel	2-17
Preparing the Netlist Hierarchy for Optimal Physical Partitioning	2-18
Saving the Floorplan Information	2-20
Writing Floorplan Physical Constraints for Design	
Compiler Topographical Mode	2-22
Defining the Physical Constraints	2-23
3. Automating Die Size Exploration	
Preparing the Design for MinChip Technology in IC Compiler	3-4
Using the Estimate Area GUI	3-8
Using the estimate_fp_area Command and Options	3-18
Using Multivoltage Designs in MinChip Technology	3-25
4. Handling Black Boxes	
Reading Netlists With Black Boxes	4-2
Creating CEL Views for Black Boxes	4-3
Sizing Black Boxes by Gate Equivalence	4-5
Estimating Black Box Sizes	4-5
Managing the Attributes for a Black Box	4-7
Flattening Existing Black Box Instances	4-7
Creating Quick Timing Models for Black Boxes	4-7
Using the Quick Timing Model Command Flow for Black Boxes	4-8
Prerequisites for Running the Quick Timing Model Flow	4-9
Creating a Quick Timing Model for Black Boxes	4-10
Quick Timing Model Example of a Black Box	4-12

5. Performing an Initial Virtual Flat Placement

Evaluating Initial Hard Macro Placement	5-2
Specifying Hard Macro Placement Constraints	5-3
Creating a User-Defined Array of Hard Macros	5-3
Setting Floorplan Placement Constraints On Macro Cells.....	5-9
Placing a Macro Cell Relative to an Anchor Object.....	5-14
Reporting Relative Location Constraints	5-15
Extracting Relative Location Constraints	5-15
Removing Relative Location Constraints	5-16
Using a Virtual Flat Placement Strategy	5-16
Creating Macro Blockages for Hard Macros	5-25
Padding the Hard Macros	5-26
Automatic Padding	5-27
Placing Hard Macros and Standard Cells.....	5-28
Planning the Location and Shape of Voltage Areas	5-28
Handling Nested Voltage Areas	5-31
Supported Voltage Area Physical Boundary Scenarios	5-32
Updating Voltage Areas in a Design.....	5-33
Removing Voltage Areas	5-33
Controlling the Placement	5-33
Moving Cells In the Core Area to a New Location.....	5-36
Packing Hard Macros Into an Area	5-37
Manually Adjusting the Hard Macros	5-38
Using the Align Objects Button.....	5-38
Using the Distribute Objects Button	5-38
Performing Simultaneous Placement and Pin Assignment for Block Level Designs.....	5-39
Supporting Relative Placement Groups in Initial Virtual Flat Placement	5-40
Using the create_fp_placement Command to Place Cells in Relative Placement Groups	5-41
Default Relative Placement Flow	5-41
Using the Relative Placement Cells Flow	5-42
Placing Multiply Instantiated Modules	5-43
Determining Which Plan Groups and Soft Macros are Multiply Instantiated Modules	5-44
Removing the Multiply Instantiated Module Property	5-44
Identifying Multiply Instantiated Module Plan Groups	5-45

Shaping the Multiply Instantiated Module Plan Groups	5-45
Placing and Analyzing the Multiply Instantiated Module Plan Groups	5-46
Copying the Cell Placement of a Plan Group	5-47
Copying Placement Blockages and Boundaries to Other Plan Groups . . .	5-48
Restoring the Placement of Cells in Plan Groups	5-48
Flipping the Placement of Multiply Instantiated Module Plan Groups	5-49
Using the Multiple Instantiated Module GUI	5-50
Performing a QoR Analysis of Multiply Instantiated Module Plan Groups	5-50
Generating a QoR Placement Report	5-51
Checking the Total Wire Length Estimate	5-51
Performing Floorplan Editing	5-51
Using Constraints With the Core and Die Editing Commands	5-52
Enabling the Feasibility Flow for Optimizing Dirty Constraint Data	5-53
Recommended Feasibility Optimization Flow	5-54
Creating a Collection of Pins	5-55
Reporting the Feasibility Options	5-56
6. Creating and Shaping Plan Groups	
Creating Plan Groups	6-2
Removing the Plan Groups	6-4
Analyzing and Manipulating the Hierarchy	6-4
Opening the Hierarchy Browser	6-5
Indicating Object Types	6-5
Exploring the Hierarchical Structure	6-6
Manipulating the Hierarchy	6-6
Merging the Hierarchy	6-7
Flattening the Hierarchy	6-13
Automatically Placing and Shaping Objects In a Design Core	6-14
Keeping the Top-level Area of the Design Contiguous During Shaping	6-18
Using Variables or Parameters to Control Channel Resizing	6-20
Using Relative Placement Constraints to Guide the Placement and Shaping of Objects	6-22
Enabling Plan Group-Aware Scan Chain Reordering	6-23
Using Plan Group-Aware Scan Reordering	6-24
Checking the Consistency of the SCANDEF Data Against the Netlist	6-26

Using Plan Group-Aware Repartitioning	6-26
Adding Padding to Plan Groups	6-27
Removing the Plan Group Padding	6-28
Adding Block Shielding to Plan Groups or Soft Macros	6-29
Removing Module Block Shielding	6-30
7. Performing Power Planning	
Editing Power and Ground and Tie-Off Connections for Nonmultivoltage Designs	7-2
Performing Automatic Connections of Hierarchical Power and Ground Pins	7-6
Example: Connecting Power and Ground Pins to Power and Ground Nets	7-9
Example: Connecting Tied-Off Pins to Power and Ground Pins..	7-10
Creating a New Hierarchical PG Network or Updating an Existing PG Network	7-11
Removing Hierarchical Power and Ground Connections.....	7-13
Manually Connecting Power and Ground and Tie Nets to Pins or Ports	7-14
Example: Using the <code>connect_net</code> command for Power and Ground Nets	7-15
Performing Hierarchical Power and Ground Connections for Tie Cells	7-16
Recovering a Tie-Off Connection From a Direct PG Connection	7-16
Reporting Tie-Off Connections	7-17
Removing Tie Cells	7-17
Reporting Power and Ground Nets	7-17
Reporting Cell Power and Ground Pin Connections	7-18
Adding Power and Ground Rings	7-18
Creating Rectilinear Rings	7-22
Creating PG Rings Around a Group of Hard Macros..	7-22
Adding Power and Ground Straps	7-24
Prerouting Standard Cells.....	7-29
Setting Preroute Design Rule Checking Options	7-32
Defining a Set of Special Preroute Rules	7-34
Setting Preroute Advanced Via Rules.....	7-35
Creating Preroute Vias	7-36

Performing Low-Power Planning for Multithreshold-CMOS Designs	7-39
Performing Multithreshold-CMOS Exploration Using	
Power Network Analysis	7-41
Exploring the Placement of Multithreshold-CMOS	
Power Switching Cells in the Power Network	7-41
Inserting Power-Switching Cells for Multithreshold-CMOS Designs	7-42
Replacing Existing Header or Footer Cells	7-43
Creating Virtual Power and Ground Nets and	
Connecting the Power-Switching Cells	7-44
Optimizing the Power-Switching Cells to Meet IR Drop Targets	7-45
Performing Multithreshold-CMOS Ring Placement	7-46
Performing Power Network Analysis on Power-Switching	
Cells in Multithreshold-CMOS Designs	7-47
Automatically Connecting Switch Pins	7-48
Performing Power Network Synthesis On Multivoltage Designs	7-50
Performing Power Network Synthesis for Power-Gating	
Designs With Multithreshold-CMOS Switch Cells	7-51
Synthesizing the Power Network for a Power-Down Voltage Area	7-51
Performing Automatic Power Switch Cell Synthesis	
for Power-Down Voltage Areas	7-52
Specifying Different Multithreshold-CMOS Placement Styles	7-54
Simulating the IR Drop Across the Multithreshold-CMOS switch Cells	7-59
Synthesizing the Common Power Network	7-60
Advanced Strategy to Control Aligned Permanent Power Straps	7-61
Defining Multivoltage Power Network Synthesis Constraints	
in a Specified Voltage Area	7-62
Removing Voltage Area Constraints	7-66
Reporting Voltage Area Constraints	7-66
Allocating Power Budgets to Each Voltage Area	7-67
Synthesizing the Voltage Areas	7-67
Generating Multi-Net Core Rings and Connecting Power Straps	7-68
Supporting Incomplete Power Network Analysis	7-69
Committing the Synthesized Results	7-69
Performing Power Network Synthesis	7-69
Why Use Power Network Synthesis?	7-70
Opening the Power Network Synthesis User Interface	7-73
Performing Power Network Stacked Via Removal to Reduce Congestion	7-77
Selecting Stacked Vias for Removal	7-78

Specifying Global Constraints for Power Planning	7-79
Specifying Core Ring and Strap Constraints for Power Planning	7-80
Specifying Constraints for Generating Multiple Core Rings or Sandwich Core Rings	7-81
Specifying Layer Constraints for Power Planning	7-82
Specifying Constraints for Block Ring Synthesis	7-83
Specifying Constraints for Power Pad Synthesis	7-84
Specifying Regions for Power Network Synthesis	7-85
Performing Region-Based Nonuniform Mesh Density Power Network Synthesis	7-86
Defining Predefined Straps Configurations	7-86
Creating User-Defined Power and Ground Routing	
Constraints in Hard and Soft Macros	7-87
Routing Constraints Format	7-87
Defining Customized Strap Configurations for Hard	
Macros and Rectangular Regions	7-88
Configuration File Format	7-88
Aligning Power Network Synthesis Straps With Top-Level Pins	7-89
Resynthesizing a Power and Ground Mesh Inside a Soft Macro	7-90
Aligning Power and Ground Straps With Standard Cell Rails	7-91
Placing Power and Ground Straps Inside Standard Cell Rows	7-91
Performing Strap Alignment With Area I/O Cells	7-92
Specifying Information in an Alignment Constraint File	7-92
Committing the Power Plan	7-93
Checking the Integrity of the Power Network	7-93
Handling TLUPlus Models in Power Network Synthesis	7-95
Specifying the Operating Temperature for IR Drop Analysis in Power Network Synthesis and Power Network Analysis	7-95
Supporting the Unified Power Format (UPF) Mode in Power Network Synthesis	7-95
Analyzing the Power Network	7-96
Performing Power Network Analysis	7-97
Adding Virtual Pads	7-100
Ignoring Hard Macro Blockages When Connecting Virtual Pads to the Power Grid	7-102
Removing Virtual Pads	7-102
Creating Virtual Connections From Power and Ground Pads to Power and Ground Wires In Power Network Analysis	7-103

Creating Connectivity (CONN) Views for Storing the Power and Ground Network of Hard Macros	7-103
Specifying the Design and Library Names	7-104
Specifying the Hierarchical Power and Ground Net Information for Extraction.	7-105
Generating Current Source Files During CONN View Generation.	7-105
Excluding Vias During CONN View Generation	7-106
Excluding Cells During CONN View and Current Source File Generation	7-106
Identifying the Geometry Annotated By the Text	7-107
Performing Power Analysis With Switching Activity Information	7-108
Annotating Switching Activity	7-108
Performing Power Network Analysis for Each Cell Instance	7-110
Handling a Long Via Array In Power Network Analysis	7-111
Viewing the Analysis Results	7-111
Displaying the Voltage Drop Map.	7-111
Displaying the Electromigration Map	7-113
Displaying the Resistance Power Map.	7-114
Displaying Instance Voltage Drop, Power, and Power Density Maps.	7-115
Displaying the Instance Voltage Drop Map.	7-115
Displaying the Instance Power Density Map.	7-116
Displaying the Instance Power Map	7-117
Summary of Power Network Analysis and Power Network Synthesis Options.	7-118
Reporting Settings for Power Network Synthesis and Power Network Analysis Strategies	7-124
Running PrimeRail Within IC Compiler	7-125
IC Compiler—PrimeRail Analysis Flow	7-125
Executing PrimeRail Commands in IC Compiler	7-126
Specifying Setup Options for Rail Analysis.	7-126
Reporting Rail Option Settings	7-128
Performing Rail Analysis	7-128
Reading and Removing Map Data Files	7-132
Debugging IC Compiler—PrimeRail Flow	7-133
Calculating Net Resistivity	7-134
Exporting PrimeRail Analysis Data for IC Compiler.	7-135
Using the <code>create_rail_setup</code> Command to Perform Reliability Analysis	7-137

8. Performing Prototype Global Routing

Setting Up for Routing	8-2
Setting Routing Options	8-2
Setting Route Guides	8-2
Setting the Preferred Routing Direction	8-4
Specifying Net Layer Constraints	8-5
Setting Nondefault Routing Rules	8-6
Setting Routing Types	8-7
Setting Net Aggressors	8-8
Running Global Prototype Routing	8-9

9. Performing Clock Planning

Setting Clock Planning Options	9-2
Reporting Clock Planning Options	9-3
Removing Clock Planning Options	9-3
Performing Clock Planning Operations	9-3
Generating Clock Tree Reports	9-4
Generating Clock Network and Source Latency for Each Clock Pin of Each Plan Group	9-5
Creating a Virtual Clock for I/O Paths	9-6
Using Multivoltage Designs in Clock Planning	9-7
Interpreting Level-Shifter Cells as Anchor Cells During Clock Planning	9-7
Performing Plan Group-Aware Clock Tree Synthesis in Clock Planning	9-8
Supporting Abutted Floorplans in Hierarchical Clock Planning	9-8

10. Performing In-Place Timing Optimization

Running In-Place Timing Optimization	10-2
Running Trace Mode In-Place Optimization	10-4
Removing the Trace Mode	10-5
Using Timing and Data Query Commands in Trace Mode In-Place Optimization	10-5
Using In-Place Optimization With Multivoltage Designs	10-6

Performing In-Place Optimizations Based on Pin Locations.....	10-6
11. Performing Routing-Based Pin Assignment	
Setting Pin Assignment Constraints	11-2
Specifying Physical Design Constraints on Pins	11-5
Honoring the Pin Physical Constraints	11-8
Reporting Pin Assignment Constraints	11-8
Setting Voltage Area Feedthrough Constraints.....	11-8
Reporting Voltage Area Constraints	11-9
Removing Voltage Area Constraints	11-9
Performing Traditional Pin Assignment.....	11-9
Performing Block Level Pin Assignment	11-12
Performing Block Level Non-Edge Pin Placement.....	11-12
Specifying Layers for Non-Edge Pin Placement.....	11-13
Checking the Non-Edge Pin Placement	11-13
Aligning Soft Macro Pins.....	11-13
Removing Soft Macro Pin Overlaps	11-16
Performing Pin Assignment on Plan Groups (Pin-Cutting Flow).....	11-17
Setting Pin Assignment Constraints	11-18
Honoring Top-Down Format Constraints	11-18
Performing Plan Group-Aware Global Routing	11-19
Detecting Buses	11-20
Detecting Clock Nets	11-20
Excluding Feedthroughs on Clock Nets	11-21
Performing Zroute-Based Plan Group-Aware Routing.....	11-21
Performing Global Routing.....	11-22
Increasing the Routing Speed	11-22
Analyzing the Pin and Feedthrough Routing	11-22
Committing the Hierarchy With Cut-Pins	11-26
Using Pin-Cutting With Multiply Instantiated Modules	11-27
Setting Pin Assignment Constraints for Multiply Instantiated Modules.....	11-28
Performing Plan Group-Aware Routing for Multiply Instantiated Modules.....	11-28
Analyzing Routes and Finalizing the Routing for Multiply Instantiated Modules	11-28

Selecting a Master Multiply Instantiated Module Plan Group	11-29
Committing a Master Multiply Instantiated Module Plan Group.....	11-30
Performing Incremental Pin-Cutting	11-31
Setting Pin Assignment Constraints on a Group of Selected Nets	11-32
Setting Pin Assignment Constraints on a Second Group of Selected Nets or on All Remaining Nets	11-32
Analyzing and Finalizing the Routing on a Separate Group of Nets	11-33
Committing the Hierarchy With Cut Pins	11-33
Creating Rectangular or Rectilinear Pin Guides.....	11-33
Accessing Information About the Pin Guides	11-35
Creating Pin Guides for Feedthroughs	11-35
Supported Feedthrough Guides	11-36
Performing Pin and Feedthrough Analysis	11-37
Removing Feedthrough Ports and Nets From Blocks and Voltage Areas.....	11-40
Refining the Pin Assignment	11-42
Checking the Pin Assignment	11-42
Checking the Pin Alignment.....	11-44

12. Performing Timing Budgeting

Timing Budgeting Prerequisites	12-2
Performing Pre-Budget Timing Analysis.....	12-3
Running the Timing Budgeter.....	12-6
Checking the QoR of the Timing Budgets	12-8
Generating a Quick Timing Model From a CEL View	12-9
Generating a Quick Timing Model for the Partitions of Large Designs	12-9
Quick Timing Model Command Summary.....	12-10
Performing Timing Budgeting On Plan Groups.....	12-11
Performing Hierarchical Signal Integrity Budgeting.....	12-13
Block-Level Hierarchical Signal Integrity Flow	12-13
Top-Level Hierarchical Signal Integrity Flow	12-14
Performing Post-Budget Timing Analysis	12-14
Performing Clock Latency Budgeting	12-15
Creating Budgets to Reflect Real Clock Latencies	12-16

Things to Look For	12-17
------------------------------	-------

13. Committing the Physical Hierarchy

Converting Plan Groups to Soft Macros	13-2
Splitting the Soft Macros	13-3
Pushing Physical Objects Down to the Soft Macro Level	13-4
Pushing Physical Objects Up to the Top Level	13-6
Handling 45 Degree Redistribution Layer (RDL) Routing	13-7
Committing the Hierarchy of Plan Groups With Unified Power Format (UPF) Power Domains	13-8
Converting Soft Macros to Plan Groups	13-8
Propagating United Power Format (UPF) Constraints	13-10
From the Soft Macro to the Top Cell	

Appendix A. Using the Flip-Chip Flow

Preparing the Library	A-3
I/O Drivers	A-3
Bump Cells	A-3
Creating an Initial Die Size	A-5
Using a Die-Driven (IC-Driven) Driver and Bump Placement Flow	A-6
Placing the Flip-Chip I/O Drivers	A-7
Creating a Pattern of Bump Cells in a Ring Configuration	A-7
Creating a Pattern of Bump Cells in an Array Configuration	A-8
Automatically Assigning Nets From Bumps to I/O Drivers	A-9
Merging Multiple Flip-Chip Nets Into One Net	A-11
Assigning Bump Pattern Personality Types	A-12
Using a Package-Driven (Bump-Driven) Bump and Driver Placement Flow	A-13
Importing and Placing Bump Cells	A-14
Saving the Flip-Chip Bump Cells	A-16
Creating a Secondary Grid for Flip-Chip Driver Placement	A-16
Defining Legal Locations for Placing Flip-Chip Drivers	A-17
Setting Options for Flip-Chip Driver Placement	A-19
Placing Flip-Chip I/O Drivers Concurrently With Standard Cells and Hard Macros	A-20

Reporting Flip-Chip Driver to Bump Results	A-20
Performing Automatic Bump Net Routing	A-21
Using the Flip-Chip Routing Commands	A-21
Specifying Routing Rules and Options	A-22
Creating Stack Vias on Pad Pins	A-23
Performing Redistribution Layer Routing	A-23
Resolving the Routing Congestion	A-23
Optimizing the Routing Pattern	A-26
Analyzing the Routing Results	A-26
Verifying Design Rule Checking and Open Nets	A-26
Routing the Flip-Chip Nets	A-27
Using Flip-Chip Structures in Cover Macros	A-29
Summary of the Flip-Chip Commands	A-29

Index

Preface

This preface includes the following sections:

- [What's New in This Release](#)
- [About This Guide](#)
- [Customer Support](#)

What's New in This Release

Information about new features, enhancements, and changes, along with known problems and limitations and resolved Synopsys Technical Action Requests (STARs), is available in the *IC Compiler Release Notes* in SolvNet.

To see the *IC Compiler Release Notes*,

1. Go to the Download Center on SolvNet located at the following address:

<https://solvnet.synopsys.com/DownloadCenter>

If prompted, enter your user name and password. If you do not have a Synopsys user name and password, follow the instructions to register with SolvNet.

2. Select IC Compiler, and then select a release in the list that appears.
-

About This Guide

The IC Compiler tool provides a complete netlist-to-GDSII or netlist-to-clock-tree-synthesis design solution, which combines proprietary design planning, physical synthesis, clock tree synthesis, and routing for logical and physical design implementations throughout the design flow.

This guide describes the IC Compiler design planning flow; the companion volume, IC Compiler Implementation User Guide, describes the implementation and integration flow.

Audience

This user guide is for design engineers who use IC Compiler to implement designs.

To use IC Compiler, you need to be skilled in physical design and design synthesis and be familiar with the following:

- Physical design principles
 - The Linux or UNIX operating system
 - The tool command language (Tcl)
-

Related Publications

For additional information about IC Compiler, see Documentation on the Web, which is available through SolvNet at the following address:

<https://solvnet.synopsys.com/DocsOnWeb>

You might also want to refer to the documentation for the following related Synopsys products:

- Design Compiler
- Milkyway Environment

Conventions

The following conventions are used in Synopsys documentation.

Convention	Description
Courier	Indicates command syntax.
<i>Courier italic</i>	Indicates a user-defined value in Synopsys syntax, such as <i>object_name</i> . (A user-defined value that is not Synopsys syntax, such as a user-defined value in a Verilog or VHDL statement, is indicated by regular text font italic.)
Courier bold	Indicates user input—text you type verbatim—in Synopsys syntax and examples. (User input that is not Synopsys syntax, such as a user name or password you enter in a GUI, is indicated by regular text font bold.)
[]	Denotes optional parameters, such as <i>pin1 [pin2 ... pinN]</i>
	Indicates a choice among alternatives, such as low medium high (This example indicates that you can enter one of three possible values for an option: low, medium, or high.)
—	Connects terms that are read as a single term by the system, such as set_annotated_delay
Control-c	Indicates a keyboard combination, such as holding down the Control key and pressing c.
\	Indicates a continuation of a command line.
/	Indicates levels of directory structure.
Edit > Copy	Indicates a path to a menu command, such as opening the Edit menu and choosing Copy.

Customer Support

Customer support is available through SolvNet online customer support and through contacting the Synopsys Technical Support Center.

Accessing SolvNet

SolvNet includes an electronic knowledge base of technical articles and answers to frequently asked questions about Synopsys tools. SolvNet also gives you access to a wide range of Synopsys online services including software downloads, documentation on the Web, and “Enter a Call to the Support Center.”

To access SolvNet, go to the SolvNet Web page at the following address:

<https://solvnet.synopsys.com>

If prompted, enter your user name and password. If you do not have a Synopsys user name and password, follow the instructions to register with SolvNet.

If you need help using SolvNet, click HELP in the top-right menu bar or in the footer.

Contacting the Synopsys Technical Support Center

If you have problems, questions, or suggestions, you can contact the Synopsys Technical Support Center in the following ways:

- Open a call to your local support center from the Web by going to <http://solvnet.synopsys.com> (Synopsys user name and password required), and then clicking “Enter a Call to the Support Center.”
- Send an e-mail message to your local support center.
 - E-mail support_center@synopsys.com from within North America.
 - Find other local support center e-mail addresses at <http://www.synopsys.com/Support/GlobalSupportCenters/Pages>
- Telephone your local support center.
 - Call (800) 245-8005 from within the continental United States.
 - Call (650) 584-4200 from Canada.
 - Find other local support center telephone numbers at <http://www.synopsys.com/Support/GlobalSupportCenters/Pages>

1

Introduction to Design Planning

You can use design planning and chip-level analysis capabilities to handle designs of various sizes and complexities. Design planning and chip-level analysis is intended to be used for both a fast exploration of the design to reduce die size and to implement a final, optimized, and detailed floorplan.

This chapter includes the following sections:

- [Overview](#)
- [Why Plan the Design?](#)
- [Using a Hierarchical Methodology](#)

Overview

Design planning in IC Compiler provides you with basic floorplanning and prototyping capabilities on flat or hierarchical designs, such as:

- Dirty-netlist handling (flat and hierarchical)
- Automatic die size exploration (flat and hierarchical)
- Performing various operations with black box modules and cells (flat and hierarchical)
- Fast placement of macros and standard cells (flat and hierarchical)
- Packing macros into arrays (flat and hierarchical)
- Creating and shaping plan groups (hierarchical only)
- In-place optimization (flat and hierarchical)
- Prototype global routing analysis (flat and hierarchical)
- Hierarchical clock planning (hierarchical only)
- Performing pin assignment on soft macros and plan groups (hierarchical only)
- Performing timing budgeting (hierarchical only)
- Converting the hierarchy (hierarchical only)
- Refining the pin assignment (hierarchical only)

Power network synthesis, applied during the feasibility phase of design planning, provides automatic synthesis of local power structures within voltage areas. Power network analysis performs voltage-drop and electromigration analysis. Power network analysis can be applied to partial, complete, or power structures created by power network synthesis. Both power network synthesis and power network analysis can be used for single or multisupply designs. You can also perform low-power planning for multithreshold-CMOS designs.

MinChip technology in IC Compiler automates the processes of exploring and identifying the valid die areas to determine the smallest routable die size for your design while maintaining the relative placement of hard macros and I/O cells. Optionally, it uses power network synthesis for power routing that meets voltage drop requirements and routing estimation technology to access routing feasibility. The technology is integrated into the Design Planning tool through the `estimate_fp_area` command. The input is a physically flat Milkyway CEL view.

Why Plan the Design?

Design planning is an integral part of the RTL to GDSII design process. It is done to quickly assess the feasibility of implementation strategies early in the flow for both flat and hierarchical designs. Design planning has become critical for large chips because they are more likely to have long interblock paths whose delays can make timing closure difficult, leading to time consuming and unpredictable tapeout schedules. As chips migrate to technologies of 0.13 microns and below, their size and complexity makes them more vulnerable to potential timing and power problems and therefore, more in need of careful planning early in the design flow.

In the design planning context, floorplanning is the process of sizing and placing cells and blocks in a way that makes later physical design steps more effective; floorplanning in hierarchical flows also provides a basis for estimating the timing of the top-level interconnect for verification. Timing budgets are allocated to each block based on the top-level timing estimation. Floorplanning can be an iterative process in which the blocks are reshaped and replaced, the timing budgets are reallocated, and the top-level timing is rechecked until an optimal floorplan is reached.

A good, effective floorplan helps ensure timing closure in many ways, such as placing blocks such that critical paths are short, by preventing routing congestion that can lead to longer paths, and by eliminating the need for over-the-top routing for noise-sensitive blocks. The challenge is to create a floorplan with good area efficiency, to conserve silicon real estate and to leave sufficient area for routing.

Similarly, design planning supports power planning, including low power design techniques that can be used in single or multisupply designs as well as multivoltage designs and multithreshold-CMOS power switching. Multisupply designs are one style of low power design. Power network synthesis and power network analysis enable designers to create power structures that meet voltage drop and electromigration specifications while minimizing the consumption of valuable routing resources. The placement engine recognizes voltage domains and keeps the cells of voltage domains close together. Once voltage areas are defined, power network synthesis creates power meshes for all voltage areas simultaneously. Floorplanning also helps avoid IR drop and electromigration problems through strategies such as placing the most power-hungry blocks close to the periphery of the chip and preventing the concentration of blocks in any one area. (For more information, see [“Performing Power Planning” in Chapter 7.](#))

IC Compiler includes complete hierarchical design planning for both channeled and abutted layout styles.

Using a Hierarchical Methodology

Using a hierarchical design methodology provides a “divide and conquer” approach for large designs. By dividing the design into multiple blocks, sometimes referred to as subblocks, modules, or lower-level blocks, designers can work on the blocks separately and in parallel from RTL through physical implementation. Working with smaller blocks can reduce overall runtimes.

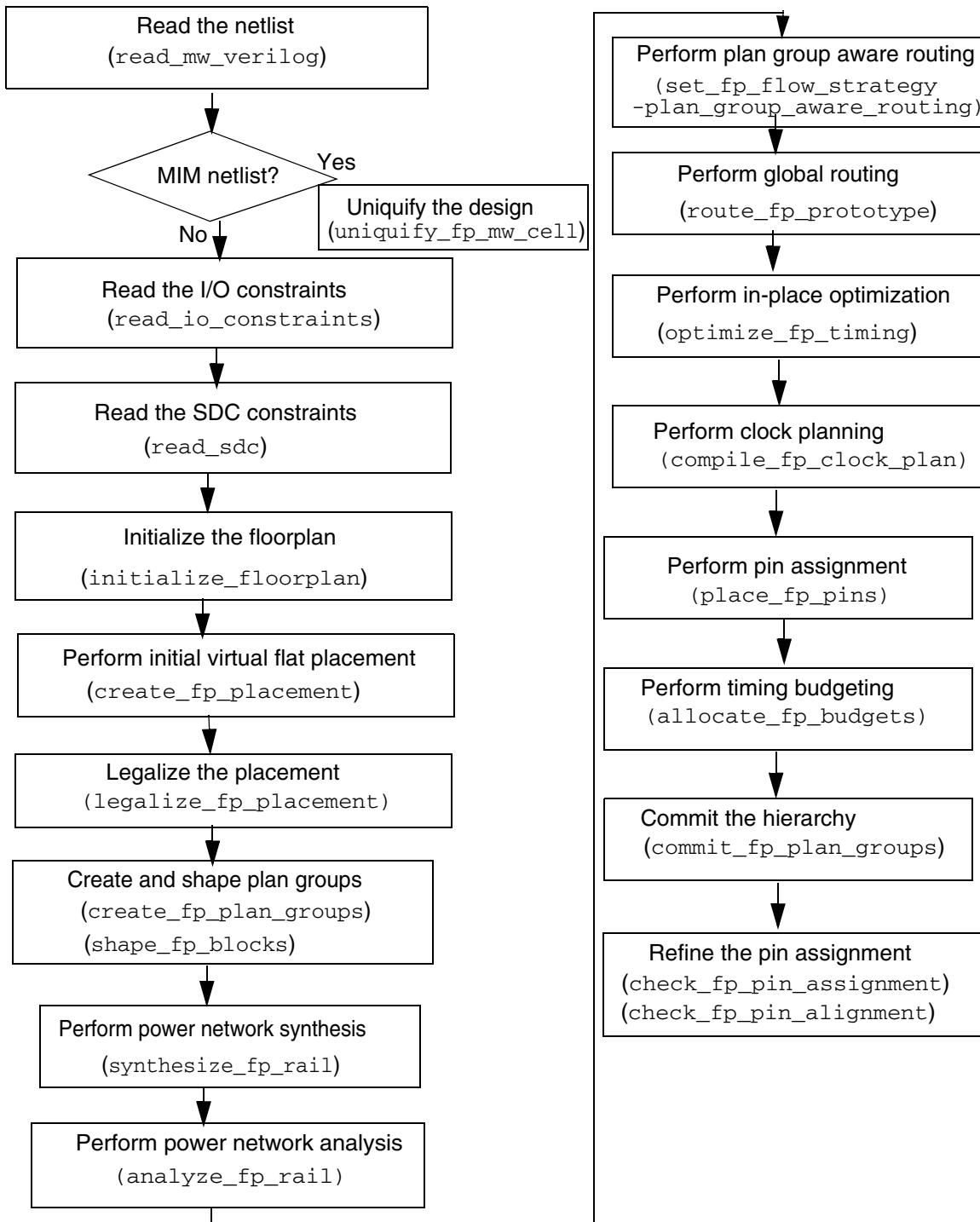
Consider using a hierarchical methodology in the following scenarios:

- The design is so large and complex that it would exceed machine’s memory capacity or runtimes.
- The design can be partitioned into subblocks that can be designed separately and worked on in parallel by different design teams.
- A number of blocks are anticipated to have problems, which might cause the schedule to slip. Using a hierarchical methodology allows late design changes to be made to individual blocks without disturbing the rest of the design.
- The design contains hard intellectual property (IP) macros such as RAMs, or the design was previously implemented and can be converted and reused.

Hierarchical Methodology for Design Planning

Once the initial design netlist is generated in Design Compiler topographical mode, you can use the hierarchical methodology for design planning in IC Compiler. Design planning is performed during the first stage of the hierarchical flow in order to partition the design into blocks, generate hierarchical physical design constraints, and allocate top-level timing budgets to lower-level physical blocks.

[Figure 1-1 on page 1-5](#) shows the hierarchical flow steps for design planning.

Figure 1-1 Hierarchical Design Planning Flow

This section includes a more detailed discussion of the following uses of design planning.

- [Design Partitioning](#)
- [Hierarchical Floorplanning](#)
- [Hierarchical Timing Closure](#)

Design Partitioning

The first step of a hierarchical design is to determine the physical partitioning. When deciding on the physical partition of a large design, you should consider the following factors:

- Size

Ensure the size of the blocks is appropriate for the virtual flat IC Compiler implementation flow. It is easier to floorplan with same-size blocks, so small blocks should be grouped and large blocks divided when appropriate.

- Data paths

Partition your design using its functional units for verification and simulation purposes. Consider top-level connectivity and minimal block pin counts to avoid congestion and timing issues.

- Floorplan style

Different floorplan styles require different physical hierarchies to support them. An abutted floorplan style has no top-level logic and a channeled floorplan has either a small or large amount of top-level logic.

- Connection with the hierarchical Design Compiler topographical mode

To exchange SCANDEF information at the block-level and the top-level, the physical hierarchy used in Design Compiler topographical mode must also be used in IC Compiler.

In the design planning stage, a physical partition is represented by a plan group. A plan group is a physical constraint for placing the cells of the same physical partition together. Each plan group represents a module in the logic hierarchy that you want to implement as a physical block and contains only cells that belong to that module. IC Compiler supports both rectangular and rectilinear plan groups. For more information, see [“Creating Plan Groups” in Chapter 6](#).

Deciding on the Physical Partitions

IC Compiler provides the following features to help you decide on the physical partitions:

- Using the Hierarchy Browser

You can use the hierarchy browser to navigate through the design hierarchy and to examine the logic design hierarchy and display information about the hierarchical cells and logic blocks in your design. You can select the hierarchical cells, leaf cells, or other objects you want to examine in layout or schematic views.

To open the hierarchy browser, select one of the following menu items:

- In the layout window, choose Partition > New Hierarchy Browser View
- In the main window, choose Hierarchy > New Hierarchy Browser View

The view window consists of two panes with an instance tree on the left and an object table on the right. The instance name of the top level design appears at the top of the instance tree.

When you create the physical hierarchy, it is recommended that you use the existing modules in the original logical hierarchy for the physical partitions. If you cannot use the original logical hierarchy, you can use the `merge_fp_hierarchy` and `flatten_fp_hierarchy` commands to modify the logical hierarchy.

For more information, see “[Analyzing and Manipulating the Hierarchy](#)” in Chapter 6.

- Performing Virtual Flat Placement

You can use virtual flat placement (the `create_fp_placement` command) to identify the logical hierarchy modules that can be used as physical partitions. If you do not know how to do the physical partitioning, you can run virtual flat placement without any partition constraints. The resulting log file contains information about the hierarchy nodes that are automatically selected for physical partitioning. The suggested physical partitioning is based on size so that the physical hierarchy is well balanced.

A hierarchical gravity feature tends to keep cells of a hierarchical design logic block physically together in the chip layout. This type of grouping usually results in better placement because designs tend to partition well along hierarchical boundaries.

For more information, see “[Placing Hard Macros and Standard Cells](#)” in Chapter 5.

- Placing and Shaping Plan Groups

Based on the distribution of cells resulting from the initial virtual flat placement, you can use the `shape_fp_blocks` command to place, shape and size plan groups automatically inside the core area. After placement and shaping, you can adjust the boundaries of the plan groups to finalize the floorplan.

For more information, see “[Automatically Placing and Shaping Objects In a Design Core](#)” in Chapter 6.

Hierarchical Floorplanning

Hierarchical floorplanning includes the following steps:

- Padding the plan groups
- Shielding plan groups or soft macros
- Assigning pins
- Creating soft macros from the plan groups
- Pushing down physical objects

Padding the Plan Groups

In the hierarchical design phase, you should add padding around plan group boundaries by using the `create_fp_plan_group_padding` command. Plan group padding sets placement blockages on the internal and external edges of the plan group boundary to prevent cells from being placed in the space around the plan group boundaries. If cells are placed too close to the plan group (later soft macros) boundaries, it may result in congestion and implementation issues.

The plan group padding is visible in the layout window. The plan group padding is dynamic, which means that it follows the changes of the plan group boundaries.

For more information, see “[Adding Padding to Plan Groups](#)” in Chapter 6.

Shielding Plan Groups or Soft Macros

To avoid congestion and crosstalk issues between a block and the top-level, you can add modular block shielding to plan groups and soft macros by using the `create_fp_block_shielding` command. The shielding creates rectangular metal layers around the outside of the soft macro boundary or plan group in the top level of the design and around the inside boundary of the soft macro or plan group, or both.

Usually, you can selectively create metal layers to block the router from routing long nets along the block boundaries. Note, however, that even if you block layers in the preferred direction, nets can still sometimes be routed along the block boundaries. In that case, you can consider blocking all layers. The block shielding is created along the soft macro boundaries, but it leaves narrow openings to access the pins.

For more information, see “[Adding Block Shielding to Plan Groups or Soft Macros](#)” in Chapter 6.

Assigning Pins

Before you can perform pin assignment, you must complete the chip-level floorplan, placement, and netlist optimization.

To perform pin assignment,

1. Identify the clock nets. Before you set the pin assignment constraints, you can ensure that there are optimal locations reserved for the pin of clock nets by using the `mark_clock_tree -clock_net` command. By doing this, pin assignment gives high priority to clock pins and avoids feedthroughs on the clock nets.
2. Set the pin assignment constraints prior to performing pin assignment by using the `set_fp_pin_constraints` command. These pin constraints are honored during pin assignment on soft macros (traditional pin assignment) and on plan groups (pin cutting flow). For more information, see “[Setting Pin Assignment Constraints](#)” in Chapter 11.
3. You can use plan group-aware global routing to suggest pin locations. To enable plan group-aware global routing, use the following command:

```
set_fp_flow_strategy -plan_group_aware_routing true
```

When plan group-aware routing is enabled, the `route_global` command honors the hierarchical rules for plan group boundary crossings by recognizing that plan groups and the routes internal to a plan group will not cross the plan group boundaries. The routes that cross the plan groups make minimal intersections with the plan group boundaries and do not crisscross. The intersection becomes the pin location of the plan groups. You can analyze the pin assignment by checking the topology of related global routes.

For more information, see “[Performing Plan Group-Aware Global Routing](#)” in Chapter 11.

4. Use the `analyze_fp_routing -finalize_pins_feedthroughs` command to save the pin locations. The pin locations for each boundary location are calculated and saved in the Milkyway database as a guide to pin placement later during commit hierarchy.

If feedthrough creation is enabled, and feedthrough nets are needed, the IC Compiler tool creates new pins for the feedthrough nets in the logical hierarchy and saves the pin locations.

If feedthroughs are created and feedthrough pins are added, you can use the `optimize_fp_timing -feedthrough_buffering_only` command to insert buffers on the feedthrough nets. This avoids assign statements in the Verilog output file and improves the timing on interblock timing paths.

Note:

After you finalize the routing, using the `analyze_fp_routing` command, any changes made to the global routes no longer affect pin assignment.

For more information, see “[Analyzing the Pin and Feedthrough Routing](#)” in Chapter 11.

5. Assign top-level or block-level pins on both rectangular and rectilinear soft macros by using the `place_fp_pins` command. The pin assignment engine considers the top-level connections to plan groups, macros, and pad locations when it determines where to assign the pins.

You can also improve pin locations based on cell placement inside the soft macros by using the `place_fp_pins -block_level` command.

For more information, see “[Performing Traditional Pin Assignment](#)” in Chapter 11.

6. Refine the pin assignment if necessary. You can analyze and evaluate the quality of the pin assignment results by checking the placement of soft macros pins in the design and by reassigning individual pins by using the `place_fp_pins` command, or you can manually change the pin assignment in the GUI.

Check or verify the placement of soft macro pins in a design. Once pins have been assigned, added, or moved, you can quickly verify if there are any spacing errors or missing pins by using the `check_fp_pin_assignment` command.

Check the pin alignment quality of results by using the `check_fp_pin_alignment` command.

For more information, see “[Refining the Pin Assignment](#)” in Chapter 11.

7. After finalizing the floorplan, you can commit the physical hierarchy by using the `commit_fp_plan_groups` command to convert the plan groups into soft macros. The command also places pins physically on the soft macro. After this, you can visually see that the pins are assigned.

For more information, see “[Converting Plan Groups to Soft Macros](#)” in Chapter 13.

Creating Soft Macros From the Plan Groups

You can convert selected plan groups or all plan groups with placement constraints into soft macro CEL views by using the `commit_fp_plan_groups` command. Each soft macro is a CEL view of the block in the same Milkyway design library. All related design information, including the netlist, floorplan, pin assignment, and power plan is saved in the CEL view. You can use the CEL view directly for block implementation in IC Compiler.

For more information, see “[Converting Plan Groups to Soft Macros](#)” in Chapter 13.

Pushing Down Physical Objects

In the hierarchical design planning phase, most of the changes that are made to the floorplan and the netlist are done at the full-chip level and inherited by the soft macros after committing the plan groups. In some cases, however, changes to your design must be made at the top level. You can do this by using the `push_down_fp_objects` command to push down the physical objects such as routing, route guides, blockages, cells, and cell rows, from the top level to the soft macro level. You can also use the `push_up_fp_objects` command to push objects up from a soft macro back up to the top level.

For more information, see “[Pushing Physical Objects Down to the Soft Macro Level](#)” in Chapter 13 and “[Pushing Physical Objects Up to the Top Level](#)” in Chapter 13.

Hierarchical Timing Closure

Timing closure in a hierarchical design is more challenging than that in a flat design. In a hierarchical design, timing optimization might not be able to see or change all the logic of the timing paths that are causing violations. Therefore, a hierarchical design needs a more sophisticated flow to achieve timing closure.

Timing closure tasks in a hierarchical design includes the following features:

- Early chip-level timing feasibility check
- Timing and signal integrity budgeting
- Clock planning

Early Chip-Level Timing Feasibility Check

You can check the timing of the entire design at various stages of the design planning flow by using the `check_timing` and `report_timing` commands. Depending on the options you set, you can get reports on valid paths for the entire design or for specific paths, unconstrained paths, and timing slack. The timing report helps evaluate why some parts of a design might not be optimized. You can also use the `check_fp_timing_environment` command to check zero wire delay timing and timing bottlenecks.

If the early timing check shows serious timing violations, you should run in-place optimization (the `optimize_fp_timing` command) to improve the timing of your design. When performing timing optimization, the command honors the plan groups defined in the hierarchical design as well as the scan chain connections from the SCANDEF. The result predicts critical paths that you will face during the implementation phase. Negative slack on interblock paths after in-place optimization should not exceed 20 percent of the clock cycle time.

For more information, see “[Running In-Place Timing Optimization](#)” in Chapter 10.

You should analyze the results, identify the root causes of the timing issues, such as the floorplan, SDC constraints, or netlist, and resolve them before proceeding to the next steps.

Timing and Signal Integrity Budgeting

After the design passes the early timing feasibility checks, you can perform timing budgeting on plan groups by using the `allocate_fp_budgets` command. The timing budgeting determines the corresponding timing boundary constraints for each top-level soft macro or plan group (block) in a design. If the timing boundary constraints for each block are met when they are implemented, the top-level timing constraints are satisfied.

Timing budgeting generates the SDC constraints and attributes for all the individual plan groups. These constraints and attributes are then transferred to the individual soft macro blocks when the hierarchy is committed using the `commit_fp_plan_groups` command. Timing budgeting can also consider crosstalk effects across soft macro boundaries when generating SDC constraints.

Timing budgeting on plan groups runs with full-chip timing. It honors pin locations and feedthrough nets assigned to the plan groups. Reasonable timing budgets result from good chip-level timing.

For more information, see “[Performing Timing Budgeting](#)” in Chapter 12.

Clock Planning

Clock planning is an optional step in the design planning flow. You can use clock planning to estimate the clock tree insertion delay and skew in a hierarchical design. It also helps you determine the optimal clock pin locations on blocks.

If you use clock planning, run it before pin assignment. Clock planning inserts anchor cells to isolate the clock trees inside the plan group from the top-level clock tree. It then runs fast clock tree synthesis for each plan group to estimate the clock skew and insertion delay. The clock tree synthesis results are annotated on floating pins defined on the anchor cells. Clock planning synthesizes the top-level clock tree to the anchor cell floating pins.

For hierarchical timing closure, clock planning is also used to generate realistic clock latency through budgeting to fix timing violations. Top-level timing violations result not only from delays on combinational logic between registers but also from clock skew between launching and capturing registers. In a hierarchical design, clock tree synthesis inside each soft macro cannot consider clock skews with the top level and other soft macros. In the top-level integration phase, clock skews among different soft macros can cause set-up and hold violations on interblock paths. It is very hard to fix these violations without changing the soft macros. To estimate the chip-level clock skew issues for block-level implementation, consider using clock planning.

For more information, see “[Performing Clock Planning](#)” in Chapter 9.

2

Initializing the Floorplan

The steps in initializing the floorplan are described in the following sections:

- [Reading the I/O Constraints](#)
- [Defining the Core and Placing the I/O Pads](#)
- [Creating Rectilinear Core Areas](#)
- [Writing I/O Constraint Information](#)
- [Reading Synopsys Design Constraints](#)
- [Adding Cell Rows](#)
- [Removing Cell Rows](#)
- [Managing Tasks in Parallel](#)
- [Preparing the Netlist Hierarchy for Optimal Physical Partitioning](#)
- [Saving the Floorplan Information](#)
- [Writing Floorplan Physical Constraints for Design Compiler Topographical Mode](#)

Reading the I/O Constraints

You place the top-level I/O pads and pins according to the constraints by specifying Tcl I/O constraints. You can specify constraints for both I/O pads and pins. The Tcl I/O constraints are stored in the Milkyway database.

This section includes the following topics:

- [Loading the Top-Level I/O Pad and Pin Constraints](#)
 - [Setting Physical Constraints on I/O Pads](#)
 - [Setting Chip-Level Constraints on I/O Pads](#)
 - [Reporting the Pin and Pad Physical Constraints](#)
 - [Removing the Pin and Pad Physical Constraints](#)
 - [Working With Multiheight I/Os](#)
 - [Adjusting the Spacing and Location of I/O Cells](#)
-

Loading the Top-Level I/O Pad and Pin Constraints

To load the top-level I/O pad and pin constraints,

1. Choose Floorplan > Read I/O Constraints.

The Read I/O Constraints dialog box appears.

Alternatively, you can use the `read_io_constraints` command.

2. In the “Input file name” box, enter the name of the Milkyway cell or plan group into which the I/O constraints are read. If you do not specify a name, the constraints are read into the current Milkyway cell.
3. Select the Append option to append the I/O constraints to the current Milkyway cell. By default, the command automatically overwrites the existing I/O constraints.
4. Click OK.

The selected I/O constraints are read into the Milkyway database (CEL).

If I/O pad cells are in the design, the layout window (CEL) shows their placement locations.

Note:

I/O pads are automatically placed according to the constraint file directions when you run the `initialize_floorplan` command.

Setting Physical Constraints on I/O Pads

Use the `set_pad_physical_constraints` command to set physical constraints on I/O pads. When you save the design, the constraints are stored in the Milkyway database. The constraints are used by the synthesis tools in IC Compiler to synthesize the design according to the specified physical constraints on the pads.

The `set_pad_physical_constraints` command uses the following syntax:

```
-pad_name  
[-side]  
[-order]  
[-offset]  
[-orientation [reflect | optimizeReflect]]  
[-min_left_iospace]  
[-min_right_iospace]  
[object]
```

Use the `-pad_name` constraint to identify the name of the pad on which the Tcl constraint is applied. You can enter valid pad names even if such a pad does not yet exist in your design. The constraints, which are stored in the Milkyway database, are applied automatically whenever a new pad with the name you have specified is added to the design.

Use the `-side` constraint to specify the edge (0 for no side; 1 for the left side; 2 for the top side; 3 for the right side; and 4 for the bottom side) of the die on which the pad is to reside. Pads are supported only for rectangular floorplans. The default is 0 (no side constraint).

Use the `-order` constraint to specify the clockwise placement order number for the pad. Enter a positive integer. A value of 0 means the pad has no ordering constraint. The default is 0.

Use the `-offset` constraint to specify the distance the pads must be placed from the edge of the die. The specified value must be a positive floating point type.

Use the `-orientation` constraint to specify if the pad instance must be reflected. The direction is derived from the Milkyway library.

Use the `-min_left_iospace` constraint to specify the minimum spacing to the next I/O pad on the left side of the die. The default value is 0.

Use the `-min_right_iospace` constraint to specify the minimum spacing to the next I/O pad on the right side of the die. The default value is 0.

Use the `-object` constraint to specify the pad to which the specified constraints apply. Currently, the command accepts only one object in the collection list. You can specify either `-pad_name` or `-object` option to set the pad constraints. These options are mutually exclusive.

See the man page for more information.

Setting Chip-Level Constraints on I/O Pads

You can use the `set_chiplevel_physical_constraints` command to specify and manage constraints on the I/O pad cells. Use this command at the chip level to specify the minimum distance in microns from which to place the I/O pads that do not directly abut the edge of the die.

The `set_chiplevel_physical_constraints` command uses the following syntax:

```
[-dist_left_edge_to_pad]
[-dist_top_edge_to_pad]
[-dist_right_edge_to_pad]
[-dist_bottom_edge_to_pad]
```

On the top and bottom edges of the die, use the `-dist_left_edge_to_pad` constraint to specify the minimum distance the I/O pads must be placed from the left edge of the die. The value must be a positive floating point type.

On the left and right edges of the die, use the `-dist_top_edge_to_pad` constraint to specify the minimum distance the I/O pads must be placed from the top edge of the die. The value must be a positive floating point type.

On the top and bottom edges of the die, use the `-dist_right_edge_to_pad` constraint to specify the minimum distance the I/O pads must be placed from the right edge of the die. The value must be a positive floating point type.

On the left and right edges of the die, use the `-dist_bottom_edge_to_pad` constraint to specify the minimum distance the I/O pads must be placed from the bottom edge of the die. The value must be a positive floating point type.

See the man page for more information.

Reporting the Pin and Pad Physical Constraints

You can use the `report_io_constraints` command display a report of the pin and pad physical constraints that were previously set by using the `set_pad_physical_constraints`, `set_chiplevel_physical_constraints`, `read_io_constraints` or `set_pin_physical_constraints` commands.

The `report_io_constraints` command uses the following syntax:

```
[-cell name]
[-pin_only]
```

```
[-pad_only]
[-chiplevel_pad_only]
[-objects]
```

If you do not specify an option, the command displays a report of all the physical constraints for all pins, pads, and chip-level pads.

See the man page for more information.

Removing the Pin and Pad Physical Constraints

You can use the `remove_io_constraints` command to remove all previously set pin and pad physical constraints in the currently active Milkyway cell or from a specified Milkyway cell for the specified pins, pads, or chip-level pads.

The `remove_io_constraints` command uses the following syntax:

```
[-cell name]
[-pin_only]
[-pad_only]
[-chiplevel_pad_only]
[-objects]
```

If you do not specify an option, the command removes all the physical constraints for all pins, pads and, chip-level pads.

See the man page for more information.

Working With Multiheight I/Os

If your design contains multiheight I/Os, you can adjust the placement of unconstrained pins and pads as well as the size of the core area and the cell boundary.

To adjust the placement of unconstrained pins and pads,

1. Choose Floorplan > Place I/O Pads.

The Place I/O Pads dialog box appears.

Alternatively, you can use the `place_io_pads` command.

2. Set the options, depending on your requirements.

- I/O style

Specify the I/O style you want to use for distributing the pads.

Net weighted optimization – Optimizes I/O pads based on net length and weight

- Center pack – Pulls pads to the center of each side
 - Distribute – Distributes pads evenly on each side without regard to the core boundary (the default)
 - Old Style – Meets the constraints with no biasing
 - Flip Chip I/O – Optimizes the flip-chip driving cell's location based on the corresponding area I/O cell
 - Distribute in core – Distributes pads evenly on each side within the area defined by the core boundary
 - Distribute and abut – Distributes pads evenly on each side without regard to the core boundary, abutting the left and right and the top and bottom pads
- Core distance
Enter the distance you want between the core and the boundary (right, left, top, and bottom). The tool uses this value to adjust the cell boundary.
 - Adjust core
Specify the method to use for determining the core area.
 - Do not change – Uses the existing core area (the default)
 - Macros and regions – Adjusts the core area to the minimum size required to enclose the macros and cell regions
 - Maintain offsets from pads – Adjusts the core area while maintaining offsets from the pads
3. Click OK or Apply.

Adjusting the Spacing and Location of I/O Cells

You can adjust the spacing and location of selected I/O cells. Only cells with the I/O pad type are moved. The I/O cells that are moved are marked as “fixed.” Any I/O-to-I/O cell overlaps are reported after the adjustment is made.

To adjust the spacing and location of selected I/O cells,

1. Choose Floorplan > Adjust I/O Placement.

The Adjust I/O Placement dialog box appears.

Alternatively, you can use the `adjust_fp_io_placement` command.

2. Set the options, depending on your requirements.

- Adjust By

Allows you to adjust the placement of I/O cells.

Side – Moves selected I/O instances or all I/O instances that are on one or more sides (left, right, top, or bottom) of the chip.

Specified I/O Cells – Enter the I/O instances before you run the `adjust_fp_io_placement` command or after the dialog box opens. If you use this option, all the selected I/O instances must already be placed on one side of the chip.

Note:

If some selected I/O instances are on one side and some are on another side, the command issues an error message and no change to the I/O placement is made.

- **Constrained By**

Allows you to specify how the spacing between adjacent I/O cells is calculated.

Spacing – Enter a floating-point number to specify exactly how many microns there should be between two adjacent I/O cells.

Pad pitch – Enter a floating-point number to specify the distance, in microns, in the x-direction from one I/O cell's center to the next I/O cell's center for the top and bottom I/O cells. Specify the distance in the y-direction for the left and right I/O cells.

- **Style**

Allows you to decide the location of all the adjusted I/O cells.

Center Pack – If both the Side and Center Pack options are selected, the center point in the x-direction for the top or bottom I/O cells will match the center point (X) of the chip area after the I/O cells are adjusted. Likewise, the center point in the y-direction for left or right will match the center point (Y) after the I/O cells are adjusted.

Moreover, if both the Specified I/O Cells and Center Pack options are selected, the center point of all the selected I/O cells after the adjustment will match the center point of these same I/O cells before the adjustment.

Offset from – Enter the distance (in microns) between the lowermost (for left and right I/Os) or the leftmost (for top and bottom I/Os) point of all the adjusted I/O cells and the bottom edge or left edge of the chip.

3. Click OK or Apply.

Defining the Core and Placing the I/O Pads

You can estimate the die size for the core area of the design, based on input control parameters such as target utilization, aspect ratio, core size, row number, and boundary, and then place the I/O pads and pins according to the constraints you have specified.

To define the core and place the I/O pads and pins,

1. Choose Floorplan > Initialize Floorplan.

The Initialize Floorplan dialog box appears.

Alternatively, you can use the `initialize_floorplan` command.

2. Specify the control parameters.

Indicate the method by which you want to specify the size of the core area:

- Aspect ratio – A ratio of height divided by width (the default)
- Width & height – The exact width and height
- Row number – A number of rows
- Boundary – A fixed size according to the boundary defined in design planning

Note:

The remaining options in the Initialize Floorplan dialog box become active or inactive, depending on the control parameter you select.

Core utilization – Enter a core utilization factor to indicate the amount of core area used for cell placement. The number is calculated as a ratio of the total cell area (for hard macros and standard cells or soft macro cells) to the core area. A core utilization of 0.8, for example, means that 80 percent of the core area is used for cell placement and 20 percent is available for routing.

Aspect ratio (H/W) – Enter the aspect ratio (height divided by width) for the core area. If you specify a ratio of 1.00, the height and width are the same and therefore the core is a square. If you specify a ratio of 3.00, the height is three times the width.

Row/core ratio – Enter a number from 0 to 1.0 to indicate the amount of channel space to provide for routing between the cell rows. The smaller the number, the more space is left between the cell rows for routing. This value should be at least as large as the core utilization value. A value of 1.0 leaves no routing channel space.

Num rows – Enter the number of cell rows that make up the core area.

Core width – Enter the width of the core area in user units. This option is available when you select “width and height” as the control parameter.

Core height – Enter the height of the core area in user units. This option is available when you select “width and height” as the control parameter.

3. Create and configure the core rows by using the following options:

Option	Explanation
Horizontal row	Select to orient the cell rows horizontally inside the core area. Deselect to orient the cell rows vertically.
Double back	Select if you want the core to contain pairs of cell rows, with every other row flipped so that pairs of adjacent rows abut top to bottom. Deselect if you want the core to contain single rows with no pairing and flipping.
Start first row	Select if you selected “Double back” and want a pairing of cell rows to start at the bottom of a horizontal core area or at the left side of a vertical core area. Deselect if you want a single row at the bottom of a horizontal core area or at the left side of a vertical area. Subsequent rows are oriented to keep power and ground lines adjacent.
Flip first row	Select if you selected “Double back” and want a flipped row at the bottom of a horizontal core area or at the left side of a vertical core area. Deselect if you want an unflipped row at the bottom of a horizontal core area or at the left side of a vertical core area.
Core to left	Enter the distance between the left side of the core and the right side of the closest pin or pad.
Core to right	Enter the distance between the right side of the core and the left side of the closest pin or pad.
Core to bottom	Enter the distance between the bottom side of the core and the bottom side of the closest pin or pad.
Core to top	Enter the distance between the top side of the core and the top side of the closest pin or pad.

Keep macro place – Select this option to keep preplaced macro cells. Deselect to remove all preplaced macro cells.

Keep std cell place – Select this option to keep preplaced standard cells. Deselect to remove all preplaced standard cells.

Min pad height – Select this option to have the tool set the core area based on the minimum pad height.

Pad limit – Select this option to have the tool place pads with no space in between.

Pin snap – Deselect this option to prevent the tool from snapping the center of pins to the center or the closest wire track.

Note:

“Pin snap” does not snap power and ground pins, fixed pins, or pins with offset constraints.

Creating Rectilinear Core Areas

You can initialize a rectilinear floorplan by using one of the predefined templates, or you can define a custom shape with the interactive region tool. The defined rectilinear area establishes the core shape and boundary into which cells will be placed.

To create a rectilinear floorplan area,

1. Select Floorplan > Initialize Rectilinear Block.

The Initialize Rectilinear Block dialog box appears.

Alternatively, you can use the `initialize_rectilinear_block` command.

2. Set the options, depending on your requirements.

- Core shape

Choose from three types of core shapes:

Template – (the default). Select this option to create a floorplan for rectilinear blocks from a fixed set of L, T, U, or cross-shaped templates. These templates are used to determine the cell boundary and shape of the core.

Note:

By default, all I/O pins are not placed; they reside at their origin. Optionally, previously placed I/O pins can be left in their current locations.

Current boundary – Select this option to use a fixed-sized rectilinear boundary that has already been defined using the template shape option.

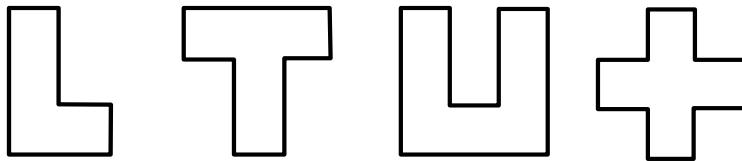
Region – Select this option to enter an interactive user-defined shape.

You can manually enter the corner-point coordinates of the rectilinear region in the Coordinates field, or you can activate the “Specify a rectilinear region” button to the right of the Coordinates field and graphically enter the corner points. The “object snap” button and the down-arrow menu button control the snapping of entered corner-points to the grid or to objects in the design layout.

- Template setup

Specify a template shape used to determine the cell boundary and core shape.

Shape – Once you have selected the “Template” core-shape mode, you must select the shape. Choose either the L shape, T shape, U shape, or the cross. The default is L.



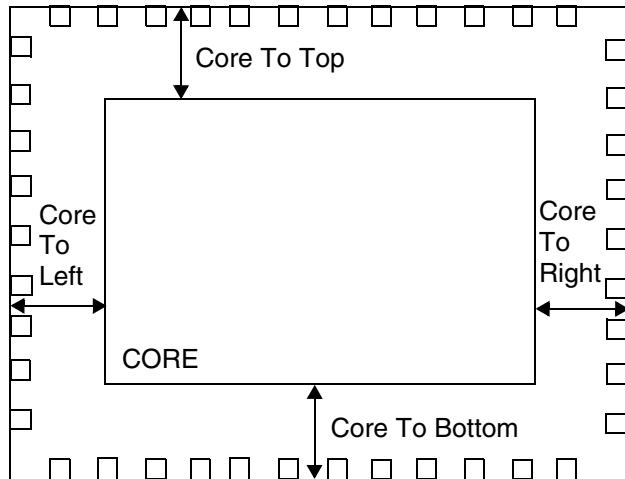
Control type – Choose the Ratio option (the default) if you want the list of dimensions (sides option) to be treated as relative dimensions to describe a ratio of proportions for each polygon side.

If you choose the length option, the dimensions are treated as actual physical dimensions for each polygon side.

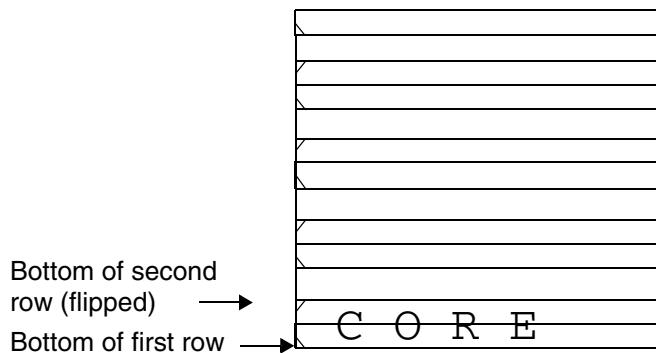
Sides – These six boxes (a through f) allow you to enter the size of the shape you selected (Shape option): the actual or proportional side length in accordance with the size control type (control option). The default is 1.0.

Orientation – Choose one of four clockwise rotations (North, West, South, or East) for the selected shape. This option redirects the positioning of the shape. North is nominal orientation, west is 90 degrees counterclockwise rotation, south is 180 degrees rotation, and east is 270 degrees counterclockwise rotation. The default is north.

- Core to left – Enter the distance between left side of the core boundary and the right side of the closest I/O pin or pad. The default is 10.0
- Core to bottom – Enter the distance between the bottom side of the core and the top side of the closest I/O pin or pad. The default is 10.0
- Core to right – Enter the distance between the right side of the core and the left side of the closest I/O pin or pad. The default is 10.0
- Core to top – Enter the distance between the top side of the core and the bottom side of the closest pin or pad. The default is 10.0.



- Core utilization – Enter a core utilization factor to enter the amount of core area to use for cell placement. The number is calculated as the ratio of the total cell area (for hard macros and standard cells or soft macro cells) to the core area. A core utilization of 0.8, for example, means that 80 percent of the core area is used for cell placement and 20 percent is reserved for routing. The default is 0.8.
- Row/core ratio – Enter a number from 0 to 1.0 to indicate the amount of channel space to provide for routing between the cell rows. The smaller the number, the more space between rows is left for routing. A value of 1.0 leaves no routing channel space. The default is 1.0.
- Horizontal rows – Select this option to orient the cell rows horizontally inside the core area. Deselect to orient the cell rows vertically.
- Double back – Select this option if you want the core area to contain pairs of cell rows, with every other row flipped so that pairs of adjacent rows abut top to bottom.

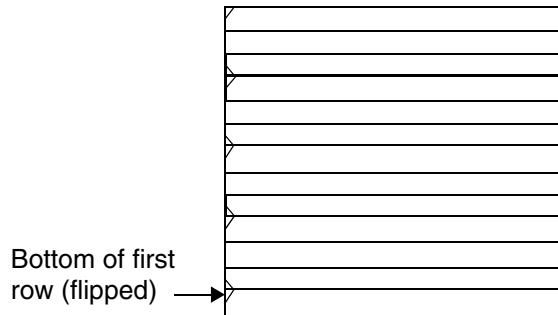


Note:

The tick mark in a corner of each row indicates the bottom of the row.

Deselect this option if you want the core to contain single rows with no pairing and flipping.

- Start first row – Select this option if you selected double back and want a pair of rows at the bottom of the core area or at the left side of a vertical core area.
Deselect this option if you want a single row at the bottom of a horizontal core area or at the left side of a vertical area. Subsequent rows are oriented to keep power and ground lines adjacent.
- Flip first row – Select this option if you selected double back and want a flipped row at the bottom of a horizontal core area or at the left side of a vertical core area.



Deselect this option if you want an unflipped row at the bottom of a horizontal core area or at the left side of a vertical core area.

- Keep placement
 - Macro cells – Select this option to keep preplaced macro cells. Deselect to remove all preplaced macro cells. This is the default.
 - Std cells – Select this option to keep preplaced standard cells. Deselect to remove all preplaced standard cells.
 - I/O terminals – When the Current boundary option is selected, select this option to keep I/O pins. Deselect to remove all I/O pins.

3. Click OK or Apply.

Writing I/O Constraint Information

You can write I/O constraint information to a specified file.

To write top-level I/O pad or pin constraints,

1. Choose Floorplan > Write I/O Constraints.

The Write I/O Constraints dialog box appears.

Alternatively, you can use the `write_io_constraints` command.

By default, the I/O constraints are written in the tcl format.

2. In the “Output file name” box, enter the name of the file to which the I/O constraints are to be written.
3. Select the “Pin only” option to write only pin constraints, or select the “I/O only” option to write only pad constraints, or “Default” to write only side and ordering information. These options are mutually exclusive.
4. Select a “Constraint type” option for the “I/O only” types of side, side and order, or side and location. These types are mutually exclusive. Select only one. The default is side and location.

Side – Writes the side constraints based on the actual locations of the specified or current Milkyway cell to the specified constraints file.

Side and Order – Writes the side and order constraints based on the actual locations of the specified or current Milkyway cell to the specified constraints file.

Side and location – Writes the side and location constraints based on the actual locations of the specified or current Milkyway cell to the specified constraints file.

5. Click OK to write the I/O constraints information of the current Milkyway cell to the specified file.

Note:

The command writes physical constraints based on the actual locations of the specified cell to a constraints file in the Tcl format. The contents of the database, where preexisting physical constraints are stored, does not change. If it is necessary to store the physical constraints in the database, based on the actual locations of the pad or pin constraints, you must first use the `write_io_constraints` command to write the constraints to a file and then use the `read_io_constraints` command to read the constraints into the database from that file.

Reading Synopsys Design Constraints

You can use the Synopsys Design Constraints (SDC) commands to do timing analysis. At a minimum, the timing constraints must contain a clock definition for each clock signal, as well as input and output arrival times for each I/O port. This requirement ensures that all signal paths are constrained for timing.

You can use a file of Synopsys Design Constraints (SDC) commands or use individual SDC commands. For details about the SDC commands, see the *Using the Synopsys Design Constraints Format Application Note*.

To read a timing constraints file, use the `read_sdc` command. You can also choose File > Import > Read SDC in the GUI.

```
icc_shell> read_sdc -version 1.7 design_name.sdc
```

After reading the SDC constraints, make sure that all paths are constrained. If the constraints are not complete, optimizations will not be performed on unconstrained paths. This can result in incorrect results after timing optimization. IC Compiler does not optimize paths that are not constrained for timing. Before proceeding, use the `check_timing` command to verify that all paths are constrained. If the `check_timing` command reports unconstrained paths, run the `report_timing_requirements` command to verify that the unconstrained paths are false paths (the `check_timing` command considers false paths unconstrained).

To remove the timing constraints set by SDC commands, use the `remove_sdc` command.

Adding Cell Rows

You can add cell rows in a specified area by entering the coordinates of diagonal corners of a rectangle. You can also create cell rows with different heights, depending on the unit tile you specify.

If you have a rectilinear floorplan, you can create additional rows with different heights in a specified area of the floorplan. IC Compiler adds the number of rows that fit in the area, based on the unit tile height and the channel height you specify.

To add cell rows,

1. Choose Floorplan > Add Site Row.

The Add Site Row dialog box appears.

Alternatively, you can use the `add_row` command.

2. Set the options, depending on your requirements.

- Double back – Select if you want the area to contain pairs of cell rows, with one row in each pair flipped.
- Start from first row – Select this option if you selected “Double back” and you want a pair of rows at the bottom of a horizontal core area or at the left side of a vertical core area. Deselect this option if you want a single row at the bottom of a horizontal core area or at the left side of a vertical core area. Subsequent rows are oriented to keep power and ground lines adjacent.
- Flip first row – Select if you selected “Double back” and want a flipped row at the bottom of a horizontal core area or at the left side of a vertical core area. Deselect if you want an unflipped row at the bottom of a horizontal core area or at the left side of a vertical core area.
- Minimum channel height – Enter the minimum amount of channel space you want to provide for routing between the cell rows.
- Tile name – Enter the name of the unit tile to be used in the cell rows. If you do not specify a unit tile name, IC Compiler uses the default tile name of “unit”.

When you set a nondefault unit tile name, it is not saved with the design; it stays in effect only until you reset it or the current session ends.

- Tile name filter – Enter the name of the tile pattern to use in the cell rows.
- Tile base – Enter the name of the tile base array in which to create the cell rows.
- Direction – Specify the direction (horizontal or vertical) of the rows to be created.
- Coordinates – Specify the diagonal corners of the area in which to create a list of rows.

3. Click OK or Apply.

Removing Cell Rows

You can remove (cut) all existing cell rows from the current design, or you can cut single cell rows in a specified area.

To remove cell rows,

1. Choose Floorplan > Cut Site Row.

The Cut Site Row dialog box appears.

Alternatively, you can use the `cut_row` command.

2. Set the options, depending on your requirements.

- All rows in design – Select this option if you want to remove all the existing cell rows.
 - Coordinates – Specify the coordinates for cutting rows from a specified area.
3. Click OK or Apply.

Managing Tasks in Parallel

You can use the `run_parallel_jobs` and `send_flow_status` commands to help manage the execution of tasks in parallel by using the Load Sharing Facility (LSF). Parallel job execution is typically used on a group of subblocks in a hierarchical design. These commands can help improve performance by allowing you to run tasks on a group of subblocks in parallel much more quickly than if they were run in serial on a single machine. By breaking up tasks into stages and running them in parallel, you can find and correct problems earlier in the design flow. Note, however, that running tasks on a set of subblocks in parallel is done at the cost of memory, as additional machines are required.

The `run_parallel_jobs` command allows you to run parallel jobs controlled by the IC Compiler parent and to communicate and return status information of the child processes to the parent. The command syntax can be used to submit jobs, to stop jobs that are currently running, and to check the status of the jobs.

The syntax to submit jobs is

```
run_parallel_jobs
[-serial | -parallel {num_jobs | all}]
[-job_name job_name]
[-stage_names stage_name_list]
[-submit_timeout timeout]
[-run_timeout timeout | -run_timeout_factor factor]
[-queue queue_name]
[-var_list name_value_pairs]
[-job_script script_path]
-batch_script script_path
-exec exec_path
-job name_list
[-verbose]
```

The syntax to stop jobs that are running is

```
run_parallel_jobs
[-kill]
[-job_name_list]
[-verbose]
```

The syntax to check the status of the jobs is

```
run_parallel_jobs
[-check]
```

You can use the `send_flow_status` command to send status information from a child job started through the `run_parallel_jobs` command back to the parent job.

The `send_flow_status` command uses the following syntax:

```
send_flow_status
[-job_name job_name]
[-host host_name]
[-port port_number]
-stage_name stage_name
-status current_status
[-eof]
[-verbose]
```

Preparing the Netlist Hierarchy for Optimal Physical Partitioning

Very large designs may not fit entirely in memory and even if they do, completing the chip implementation can take a very long time. Therefore, partitioning the design is often the best solution. You can invoke a netlist hierarchy optimizer, also referred to as a netlist partitioner by using the `optimize_netlist_hierarchy` command to identify new hierarchy automatically and to create optimal physical partitions in a design specified in the Verilog format.

The `optimize_netlist_hierarchy` command uses the following syntax:

```
optimize_netlist_hierarchy
-netlist_files filelist
-top topName
[-opaque_modules modulelist]
[-filename filename]
[-max_partition_to_chip_ratio sizeRatio]
[-keep_top_hier modulelist]
[-hier_marker symbol]
[-max_hier_mod_depth level]
[-max_num_nodes count]
[-size_metric {simple | instcnt | area}]
```

Note:

For more information about the options in this command, see the man page.

The `optimize_netlist_hierarchy` command reads a gate-level netlist design structured in the Verilog format. This input design (`-netlist_files` option) is a gate-level netlist hierarchy tree-type structure. The top module is the root of the tree, and the leaf nodes of the tree are the standard cells, hard macros, I/O pad cells, or black boxes. To create optimal physical partitions, the command identifies the hierarchy nodes in the tree-type structure that can be modified to create well-balanced physical partitions.

The `optimize_netlist_hierarchy` command outputs the following files:

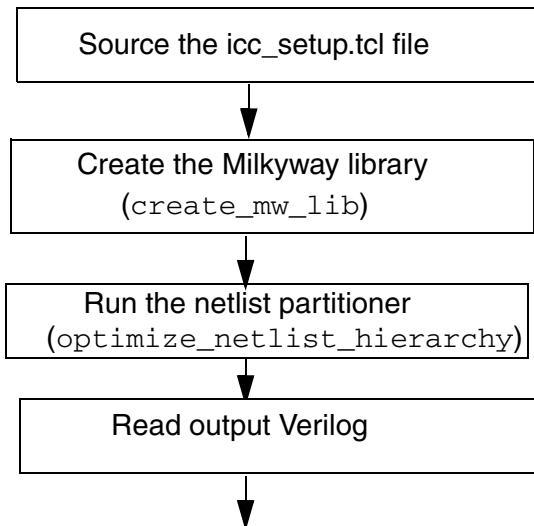
- A Verilog file (-filename option) that contains the modified hierarchical structure. The flat-level netlist in this file is functionally equivalent to the input Verilog netlist, but it does not contain the module definitions for opaque modules. The functionally equivalent netlist contains the new logical hierarchy that can be directly mapped into corresponding physical hierarchy.
- A Verilog file called `opaque_modules.v` that contains the preserved hierarchy of the design: that is, it contains the Verilog module definitions for all opaque modules.
- A Tcl command file called `partition_cmds.tcl` that contains the sequence of `ungroup` and `group` commands that when applied to the original input Verilog file, result in a partitioned netlist. This file is used primarily for generating a new timing constraints (SDC) file because the original timing constraints file might not work with a modified hierarchical netlist. The `group` and `ungroup` commands are Tcl callback functions that you must define. For example, by tying the `ungroup` command to the `flatten_fp_hierarchy` command and the `group` command to the `merge_fp_hierarchy` command, then sourcing the `partitions_cmd.tcl` file in IC Compiler, can generate a new netlist and optionally, a new SDC file.

The netlist partitioner provides the following benefits:

- It identifies an optimal partitioning scheme for a given design such that the entire chip can be divided into top-level subblock instances and various subblock masters to aid in full-chip design convergence, such that the blocks can be implemented independently and then integrated into the top level of the design.
- It ensures that the size of the partitions are balanced by either area or instance count.
- It minimizes the number of top-level nets connecting to one of more block I/O ports while honoring any user-defined constraints such as unmodifiable hierarchy or predetermined top-level only cells.
- It partitions the netlist without changing the functionality.

[Figure 2-1](#) shows the recommended netlist automatic-partitioner flow.

Figure 2-1 Recommended Netlist Automatic Partitioner Flow



Saving the Floorplan Information

You can write (save) the floorplan information from the top cell of the design, regardless of the current instance. The output is a command script file that contains information about the floorplan; therefore, you can load it as a common script file. The script file describes the floorplan information of the current Milkyway CEL view.

Note:

Loading the script file using `source floorplan.script` can be very slow. Instead, it is recommended that you use the `read_floorplan floorplan.script` to speed up the process.

You can modify this script file and use it later in the flow to re-create the floorplan.

You can write out the floorplan information for the top level of your design to a text file. You can select options for writing out specific floorplan information for basic floorplan objects, placement objects, and hierarchical floorplan objects. The output is a Tcl script file that describes the floorplan information. You can use the `read_floorplan` command to load this floorplan information in later to recreate the floorplan.

By default, floorplan information is written for the top-level design only, and it includes bounds, placement blockages, route guides, plan groups, and voltage areas.

To save the floorplan information,

1. Choose Floorplan > Write Floorplan.

The Write Floorplan dialog box appears.

Alternatively, you can use the `write_floorplan` command.

By default, the I/O constraints are written in the tcl format.

2. Next to the Output File Name box, enter the name of the output file into which the hierarchical floorplan is to be saved.
3. Top Level – Select options for writing basic floorplan information for the top level of your design.

You can write out all objects, or you can specify object types.

- Row – Includes the row information at the top level. The default is off.
- Tracks – Includes the track information at the top level. The default is off.
- Placement blockages – When this option is deselected, it prevents placement blockage information from being written to the output script. The default is on.
- Bounds – When this option is deselected, it prevents bound information from being written to the output script. The default is on.
- Plan groups – When this option is deselected, it prevents plan group information from being written to the output script. The default is on.
- Voltage areas – When this option is deselected, it prevents voltage area information from being written to the output script. The default is on.
- Route guides – When this option is deselected, it prevents route guide information from being written to the output script. The default is on.
- Preroutes – When this option is selected, preroute information is included in the output script. The default is off.
- Boundary – When this option is deselected, it prevents boundary information from being written to the output script. The default is on.

4. Hierarchical – Select options for writing out specific floorplan information for hierarchical floorplan objects in your design.

You can write out all objects, or you can specify object types.

- Cell rows – You can write specified cell row information at the subblock level for designs containing at least 1 soft macro. By default, no cell row information is written for soft macros.
- Tracks – You can write specified track information at the subblock level for designs containing at least 1 soft macro. By default, no track information is written for soft macros.

- Placement blockages – You can write specified placement blockage information at the subblock level for designs containing at least 1 soft macro. By default, no placement blockage information is written for soft macros.
 - Bounds – You can write specified bound information at the subblock level for designs containing at least 1 soft macro. By default, no bound information is written for soft macros.
 - Plan groups – You can write specified plan group information at the subblock level for designs containing at least 1 soft macro. By default, no plan group information is written for soft macros.
 - Voltage areas – You can write specified voltage area information at the subblock level for designs containing at least 1 soft macro. By default, no voltage area information is written for soft macros.
 - Route guides – You can write specified route guide information at the subblock level for designs containing at least 1 soft macro. By default, no route guide information is written for soft macros.
 - Preroutes – You can write specified preroute information at the subblock level for designs containing at least 1 soft macro. By default, no preroute information is written for soft macros.
5. Placement – Select options for writing placement information about standard cell and macro cell placement and about placement of I/O pads and pins.
- I/O pads and pins – When this option is selected, I/O pad (or terminal) and pin placement information is written to the output script.
 - Standard cell – When this option is selected, standard cell placement information is written at the top level.
 - Hard macro cells – When this option is selected, hard macro cell placement information is written at the top level.
 - Soft macro cells – When this option is selected, soft macro cell placement information is written at the top level.

Writing Floorplan Physical Constraints for Design Compiler Topographical Mode

You use IC Compiler floorplan physical constraints in the Design Compiler topographical mode to accurately represent the placement area and to improve timing correlation with the post-place-and-route design. Also, using floorplan information is recommended if the floorplan is very complicated, for example, if the design contains many macros or a large number of blockages and keepout regions or if the core area contains unusual shapes.

To extract floorplan information from IC Compiler for use in Design Compiler, use the `write_def` command in IC Compiler. This command extracts physical design data and writes this data to a Design Exchange Format (DEF) file. [Example 2-1](#) uses the `write_def` command to write physical design data to the `my_physical_data.def` file.

Example 2-1 Using write_def to Extract Physical Data From IC Compiler

```
icc_shell> write_def -version 5.6 -rows_tracks_gcells -macro -pins \
-blockages -specialnets -vias -regions_groups -verbose \
-output my_physical_data.def
```

For details about the `write_def` command, see the man page.

Note that IC Compiler also supports the `write_physical_constraints` command, but for best results, use the `write_def` command because `write_def` exports more physical information. The next section describes the commands used with the `write_physical_constraints` command.

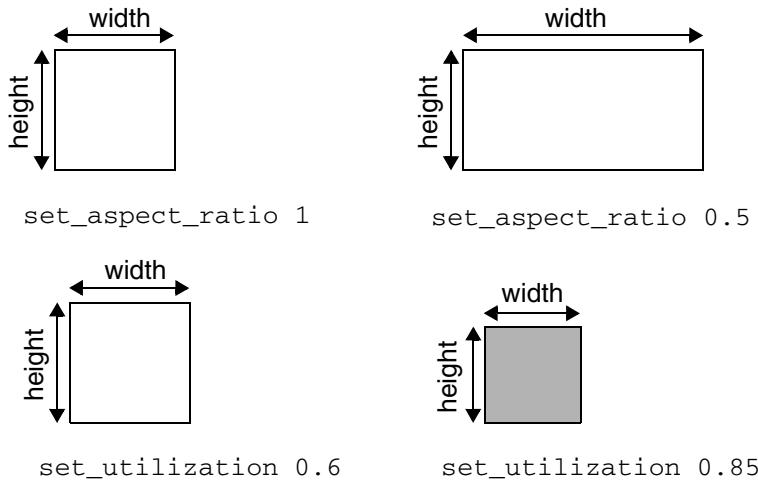
Defining the Physical Constraints

Only the high-level physical constraints that determine core area and shape, port location, macro location and orientation, and placement blockages for floorplanning are written to the output Tcl script file. These constraints are defined by the following commands:

- Core area and shape

Use the `set_aspect_ratio` and `set_utilization` commands to define the relative core area and shape as shown in [Figure 2-2](#).

Figure 2-2 Defining Relative Core Area and Shape



Note:

For multivoltage designs, you define core area and shape by using the `set_placement_area` command.

- **Exact Rectangular Core Area**

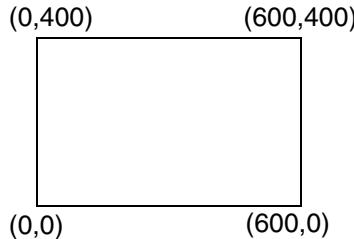
The exact rectangular block shape is passed to Design Compiler topographical mode by setting the following constraint:

```
set_placement_area -coordinate {X1 Y1 X2 Y2}
```

where `{X1 Y1 X2 Y2}` specifies the lower left and upper right coordinates of the core area.

[Figure 2-3](#) shows how you define exact core area and shape by using the `set_placement_area` command

Figure 2-3 Defining Exact Rectangular Core Area And Shape



```
set_placement_area -coordinate {0 0 600 400}
```

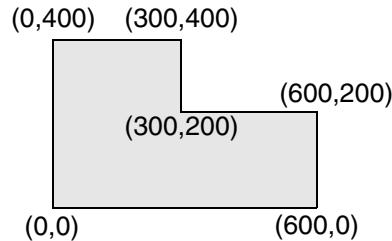
- **Exact Rectilinear Core Area**

The exact rectilinear block shape is passed to Design Compiler topographical mode by setting the following constraint:

```
set_rectilinear_outline -coordinate {X1 Y1 X2 Y2 ...}
```

where the points of the rectilinear shape are not closed; that is, the first and last points are not the same, and the points are in counterclockwise order.

[Figure 2-4 on page 2-25](#) shows how you define the exact rectilinear core area and shape by using the `set_rectilinear_outline` command.

Figure 2-4 Defining Exact Rectilinear Core Area And Shape

```
set_rectilinear_outline -coordinate {0 0 600 0 600 200 300 200 300 300 400 0 400}
```

- Port locations

The exact port locations are passed to Design Compiler topographical mode by setting the following constraint:

- `set_port_location port_name -coordinate {x y}`
where the `-coordinate` is the lower left point of the port shape.

The following example annotates port Z, a top-level port in the current design, with the location at 100, 5000.

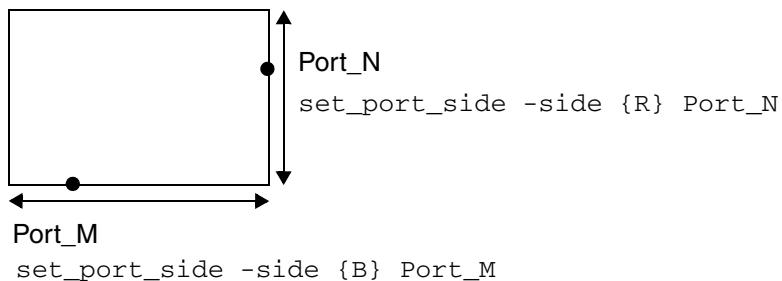
```
set_port_location -coord {100 5000} Z
```

The relative port locations are passed to Design Compiler topographical mode by setting the following constraint:

- `set_port_side port_name -side {L|R|T|B}`

This constraint is used only for a design with a rectangular core area. Valid sides are left (L), right (R), top (T), or bottom (B). A port can be placed at any location along the specified side. If the port side constraints are provided, the ports are snapped to the specified side. Otherwise, by default, the ports are snapped to the side nearest to the port location assigned by the coarse placer. Only Milkyway port locations are written out. I/O pad locations are not written out, even if the design has I/O pad cells.

[Figure 2-5](#) shows how you define port sides by using the `set_port_side` command.

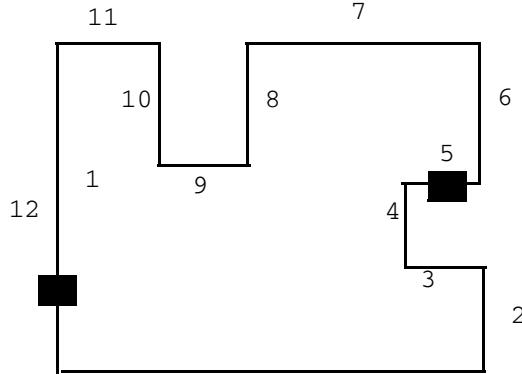
Figure 2-5 Setting Relative Port Sides

You can also use this constraint in conjunction with the `set_rectilinear_outline` command to specify a side on which the port is placed. [Figure 2-6](#) shows an example.

Figure 2-6 Setting Relative Port Sides for a Rectilinear Core

```
set_rectilinear_outline -coordinate
{0 0 100 0 100 150 75 150 75 200 100 200 100 300 40 300 40 200 30 200 30
30 0 300}

set_port_side -side 5 portA
set_port_side -side 12 portB
```



- Macro location and orientation

The exact macro locations are passed to Design Compiler topographical mode by setting the following constraint:

- `set_cell_location cell_name -coordinate {x y} -orientation {N|S|E|W|FN|FS|FE|FW} -fixed`

where `-coordinate` is the coordinate of the lower left corner of the cell's bounding box. The coordinate numbers are displayed in microns relative to the block orientation. The orientation value is one of the rotations listed in the following table.

Orientation	Rotation
N (default)	Nominal orientation, 0 degree rotation (north)
S	180 degrees (south)
E	270 degrees counterclockwise (east)
W	90 degrees counterclockwise (west)
FN	Reflection in the y-axis (flipped north)
FS	180 degrees, followed by reflection in the y-axis (flipped south)

Orientation	Rotation
FE	270 degrees counterclockwise, followed by reflection in the y-axis (flipped east)
FW	90 degrees counterclockwise, followed by reflection in the y-axis (flipped west)

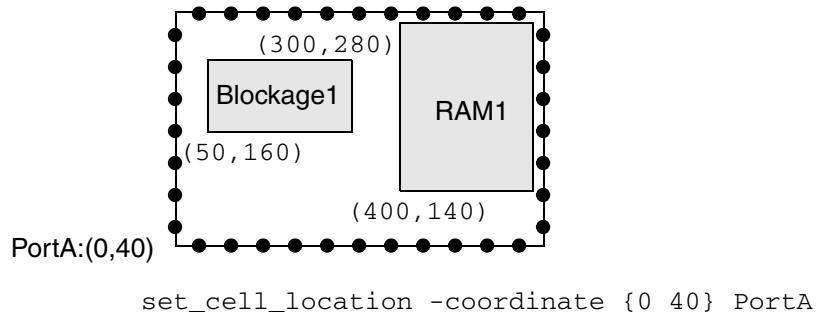
- Placement keepout

The exact placement keepouts (blockages) are passed to Design Compiler topographical mode by setting the following constraint.

- create_placement_keepout -name name_string -coordinate {llx lly urx ury} -type {soft|hard}
- where the `-name name_string` specifies the name of the keepout (blockage) and the `-type` option specifies the type of keepout to be created (soft or hard).
- Figure 2-7** shows how you use the `create_placement_keepout` command to define placement blockages.

Figure 2-7 Defining Exact Ports, Macros, and Blockages

```
create_placement_keepout\
-name BLockage1\
-coordinate {50 160 300 280}          set_cell_location\
                                         -coordinate {400 140}\\
                                         -orientation {N} -fixed RAM1
```



3

Automating Die Size Exploration

This chapter describes how to use MinChip technology in IC Compiler. Given a design that is floorplanned and placed, MinChip technology automates the processes of exploring and identifying the valid die areas to determine the smallest routable die size for your design while maintaining the relative placement of hard macros and I/O cells. Optionally, it uses power network synthesis for power routing that meets voltage drop requirements and routing estimation technology to access routing feasibility. The technology is integrated into the Design Planning tool through the `estimate_fp_area` command. The input is a physically flat Milkyway CEL view.

The `estimate_fp_area` command intelligently iterates through different design size estimates to find the smallest die or block size that meets the routability target. If the input CEL view (floorplan) is easy to route, the `estimate_fp_area` command finds the smallest routable die size. If the initial floorplan cannot be routed, the `estimate_fp_area` command produces a routable floorplan with a larger core area.

By using the MinChip technology, floorplan sizing iterations can be completed in hours, returning a result that represents the minimum area in which the design can be implemented and remain routable while retaining the characteristics of the original floorplan.

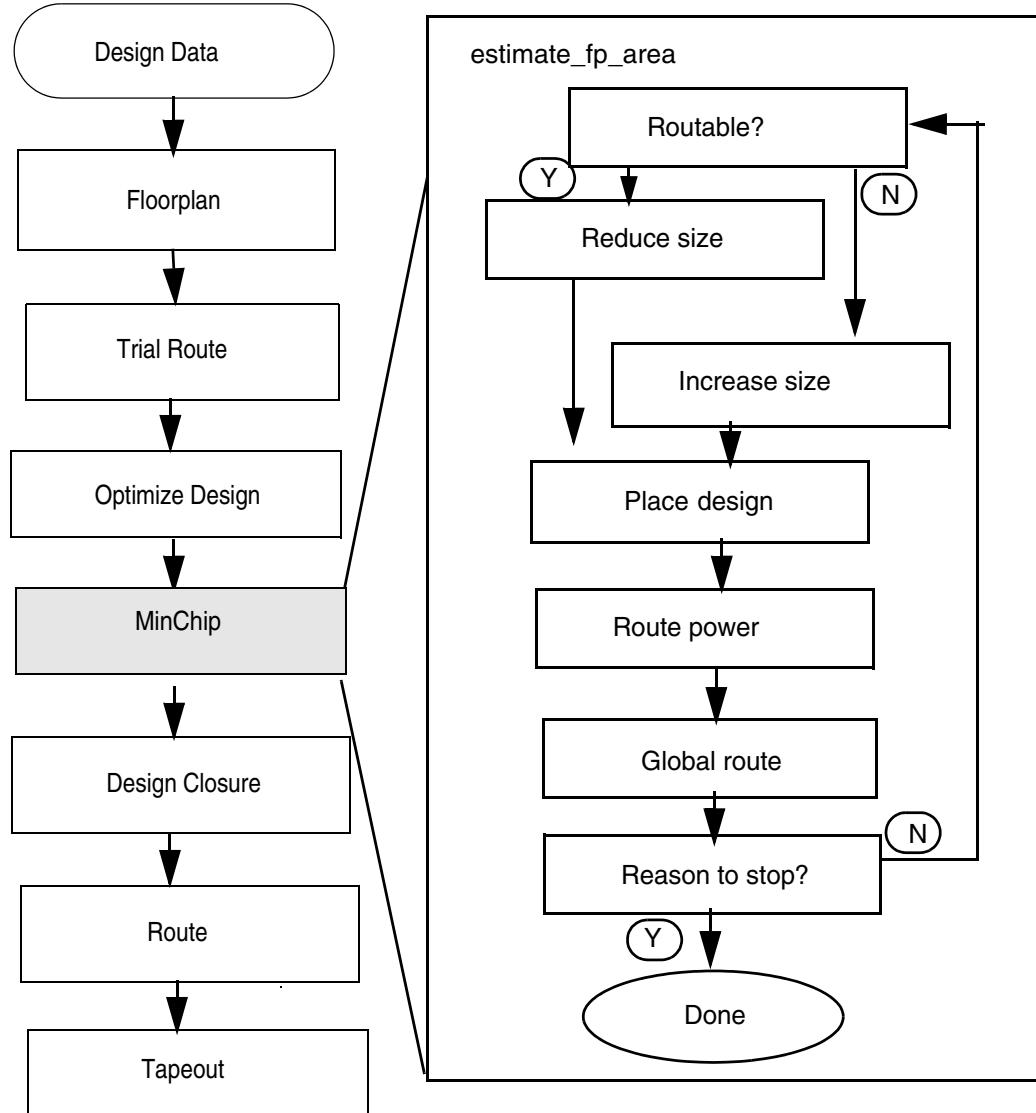
This chapter includes the following sections:

- [Preparing the Design for MinChip Technology in IC Compiler](#)
- [Using the Estimate Area GUI](#)
- [Using the `estimate_fp_area` Command and Options](#)

- Using Multivoltage Designs in MinChip Technology

Figure 3-1 shows design flow using MinChip, and the steps that occur inside the MinChip flow.

Figure 3-1 MinChip Technology Flow and Steps



The `estimate_fp_area` command performs the following steps inside MinChip:

- Estimates the core size and shape for a given core/cell utilization.

During the search for a minimal die size, filler pads and filler cells are automatically deleted to minimize the pad-limited characteristics of the design. When the size of the die changes, any previous placement blockages, preroutes, and routing guides are also automatically deleted.

To ensure that the results are consistent from each die size change, the following are maintained as constants:

- The original side and placement order of the I/O pads on the input design cell
- The relative placement of the hard macros
- The shape (aspect ratio) of the core area

2. Places all the design cells.

For a given core size, it runs virtual flat placement in incremental mode to place all the design cells.

3. Routes the design.

After cell placement, it automatically routes the design as follows:

- Performs a fast, virtual route. It estimates the wire length and calculates the routing resources to identify designs that are easy to route.
- Performs prototype global routing until the specified global route cell overflow percentage is reached. It quits routing when the design is “unroutable.”
- Performs prototype global routing to report that the design is “unroutable” when local hot spots cause a very long runtime.

Note:

In addition to the global router, MinChip supports the Zroute router. (See [“Enabling Zroute in MinChip” on page 3-8](#) or [page 3-18](#).)

Using the Zroute router in MinChip can improve the run time, provide a better reduction in the die size, and give you a more accurate prediction of the routability of the floorplan.

4. Analyzes the routing.

After the design is routed, routing analysis is done. It reports the number and percentage of nets routed, and the overall global route cell overflow percentage when a design is “unroutable.”

5. The `estimate_fp_area` command stops when any of the following occur:

- The routing target is met.
- The utilization bounds are exceeded.
- Overlaps of hard macros would be formed.

- The IR drop is exceeded.
- The design is pad limited.

Preparing the Design for MinChip Technology in IC Compiler

Because MinChip does exactly what you specify based on the options you select, it is important to understand the nuances of your design, so that MinChip can achieve the best reduction in die size in the fastest time possible. The quality of the results depends on the input CEL view and the `estimate_fp_area` command options you select.

For the best chip size reduction, make sure that your design is routable, can achieve timing closure, and the netlist is stable with no major logic changes expected in the gate-level netlist.

Note:

Applying MinChip technology to netlists and floorplans that are not implementation proven is not recommended as the results can be misleading. If the final netlist is larger than the netlist input to MinChip, it may not fit.

This section describes how to prepare your design to achieve the best QoR from MinChip.

- Floorplan

- Use a post-optimized design that has gone through initial detailed implementation in IC Compiler. MinChip supports full-chip, rectangular block, and rectilinear block floorplans.
- Correct all design rule violations.
- Verify the cause of any routing congestion. Routing congestion or violations may exist because of floorplan issues, such as macros overlapping other macros, or existing power routing that makes pins inaccessible. MinChip reduces the size of routable designs and increases the size of unroutable designs.

- Power network synthesis scripts

Power network synthesis is used in MinChip primarily to consume routing resources used by power routing. This ensures an accurate view of the available routing resources.

To create an accurate power mesh and rings, use `-run_pns_script`.

- To estimate the power network, use the power network synthesis script file that you developed during detailed implementation of the floorplan.

- If a custom power structure is required, create a power network synthesis script by using the `synthesize_fp_rail` command to emulate the power network. It mimics the custom power structure by specifying the width, pitch, and offset of the core rings, block rings, and power straps. The objective is to accurately consume the routing resources that are required by the power network.

Run power network synthesis on the existing floorplan before running the `estimate_fp_area` command to ensure that all the power network synthesis options are set properly. To run power network synthesis using the defaults, use `-power_net_names` and `-ir_drop_value`.

- Preroute standard cell scripts

Use a preroute standard cells script to preroute to the supply pins of standard cells. Do this after you run power network synthesis (`synthesize_fp_rail` command). This is important if power network synthesis is used to form a mesh on the uppermost layers. Stacked vias used to connect the metal1 preroutes to the upper layer mesh are formed during the preroute process.

To run the preroute script within the `estimate_fp_area` command, specify the script name using the `-run_preroute_script` option. Do this so that metal1 routing resources as well as the stacked vias are taken into account during die size reduction.

- Relative hard macro placement

The `estimate_fp_area` command maintains the relative placement of all fixed macros in terms of alignment and location. When a chip size is reduced, the locations of its macro cells are automatically adjusted to the relative locations. The relative horizontal and vertical ordering of the macros is maintained. You can also reduce the die or core size until the fixed macros abut.

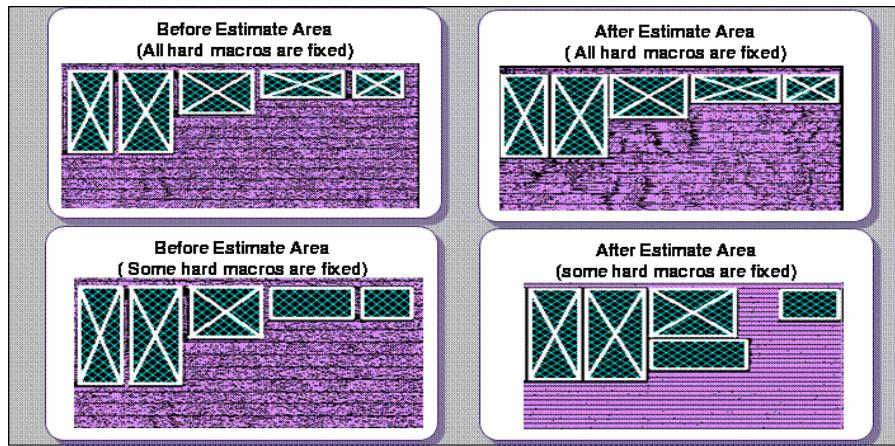
The `estimate_fp_area` command attempts to maintain the relative placement of unfixed macros, but that relative placement cannot be guaranteed. Macros that are not fixed can move to nearby locations.

Fix all macros whose placement is critical, especially any analog blocks or other macros partially outside the core area.

If there are fixed placed standard cells in your floorplan, the `estimate_fp_area` command might put hard macros on top of them. Unfix all standard cells before running the command.

[Figure 3-2 on page 3-6](#) shows before and after examples of maintaining relative hard macro placement.

Figure 3-2 Maintaining Relative Hard Macro Placement



- Hard macro padding

The `estimate_fp_area` command maintains the hard macro padding set by the `set_keepout_margin` command.

- By default, IC Compiler calculates the padding for hard macros. You can set specific values for padding by instance or master name. It is also possible, but not recommended, to set the padding to zero.
- Although macro padding resembles a halo blockage around a macro, you can specify different values for different edges.
- If the padding of adjacent macros abuts or overlaps before the `estimate_fp_area` command is run, the macros can be packed closer together.
- If the padding of adjacent macros does not overlap before you run the `estimate_fp_area` command, then it will not overlap after running the `estimate_fp_area` command.
- If the macro padding overlaps with other padding before you run the `estimate_fp_area` command, it will still overlap other padding after the `estimate_fp_area` command is run, but it will not overlap another macro.

- Sliver size

The `-sliver_size` option defines a “critical” distance for the `estimate_fp_area` command. If the distance between two objects (core edges, blockages, or macros) is less than or equal to the `sliver_size`, before running the `estimate_fp_area` command, then this distance is not reduced by the `estimate_fp_area` command.

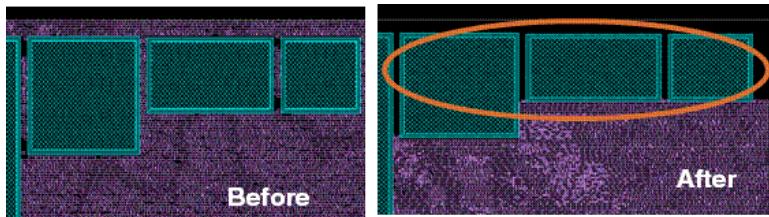
The default value for the `sliver_size` option is 0.

Analyze your floorplanned design to determine an appropriate value for the `sliver_size` and, before running the `estimate_fp_area` command, set the following parameter.

```
set_fp_placement_strategy -sliver_size distance
```

If the distance between two hard macros is less than or equal to the sliver size, the distance is honored by the `set_fp_placement_strategy -sliver_size distance` parameter, as shown in [Figure 3-3](#).

Figure 3-3 Sliver Size Packing Limit



- Placement blockages

The `estimate_fp_area` command honors both soft and hard blockages. Blockages may be assigned to an edge or macro based on the relationship between the blockage and the edge or macro. You should delete all non-essential blockages. If the intent of a blockage is to reserve a space around a macro, delete the blockage because it is nonessential (macro padding performs this function) and define the appropriate macro padding or `sliver_size`.

- Edge blockages

A blockage that covers any side of a core area is called an “edge blockage.” Edge blockages must extend over the complete edge dimension. In some cases, edge blockages are used to account for implementation requirements. For example, consider a design that requires an M1/M2 core ring as part of the power mesh. Within the `estimate_fp_area`, the power structure is formed after placement. If the core rings are likely to cover the placement area, a problem exists. Standard cells may have been placed in these areas. Edge blockages can reserve the placement area for the rings. During the `estimate_fp_area` iterations, the relative placement of edge blockages is maintained in relation to the core area.

- Blockages over fixed hard macros

Macro blockages are placement blockages assigned to macros. If macro blockages cover a macro completely or if at least 70 percent or more of the blockage area covers a macro, then the `estimate_fp_area` command assigns the blockage to the macro. During die size reduction, macro blockages move along with assigned macros. If a blockage covers two or more macros, it is randomly assigned to one macro.

If macro padding cannot be used, you should create a hard macro blockage over each macro rather than one blockage over a set of macros.

- Filler cells

The `estimate_fp_area` command deletes I/O cells marked as “`type = filler`”. Some I/O cell libraries have pads that are marked this way. To ensure that PADS are not deleted, change these cell types before you run the `estimate_fp_area` command.

Note:

If the design has standard cells marked as `type = filler`, you must explicitly delete these before running the `estimate_fp_area` command. The `estimate_fp_area` command does not automatically delete standard cell fillers in the core.

Using the Estimate Area GUI

You can use the Estimate Area GUI to automate the placement and routing iterations targeted at finding the smallest routable die size. It automatically maintains relative placement, optionally uses power network synthesis for power routing, and uses routing congestion estimation technology to assess routing feasibility.

Enabling Zroute in MinChip

In addition to the classic global router, the `estimate_fp_area` command supports the Zroute router. To enable Zroute in MinChip, set the following variable before running the `estimate_fp_area` command.

```
set_route_mode_options -zroute true
```

To explore valid die areas to determine the smallest routable die size,

1. Choose Floorplan > Estimate Area.

The Estimate Area dialog box appears.

Alternatively, you can use the `estimate_fp_area` command.

The Estimate Area dialog box options are grouped into three categories:

- Floorplan Control
- Search Control
- Power Network Control

2. Set the floorplan control options, depending on your requirements.

Note:

If you run Estimate Area with the default values (select Default), shrinkage (die size reduction) is observed in the width and height of the die until one of the search control criteria is achieved. Blockages are also removed during the default run.

- Select a sizing type to define constraints on the search space of aspect ratios that are used to shrink the die.

Variable aspect ratio – The default ratio that varies the search space of aspect ratios.

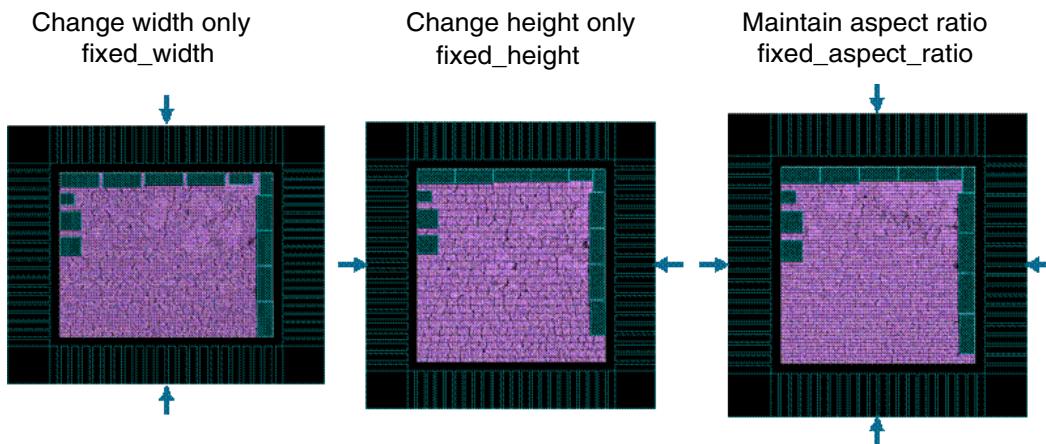
Fixed aspect ratio – Honors the aspect ratio of the original design size and stops shrinking the width or height of the die when the width or height becomes pad or macro limited. By default, estimate area ignores the aspect ratio of the input design and continues to shrink the core size to the smallest size.

Fixed width – Preserves the original design width and performs and reduces the chip size in the y-direction only. While reducing the chip size in the y-direction, the macros can be abutted if any of the stop criteria you have specified have not been reached yet.

Fixed height – Preserves the original design height and reduces the chip size in the x-direction only. While reducing the chip size in the x-direction, the relative macro locations and orientations are maintained. The chip size reduction continues until any one of the stop criteria you have specified is met.

[Figure 3-4](#) shows an example of controlling the chip sizing.

Figure 3-4 Controlling the Chip Size



- Specify the core area boundaries.

Minimum width – Enter the lower bound of the core width in microns to stop the die search process. By default, no die sizes that result in utilizations greater than 0.95 (95 percent) are considered.

Minimum height – Enter the lower bound of the core height in microns to stop the die search process. By default, no die sizes that result in utilizations greater than 0.95 (95 percent) are considered.

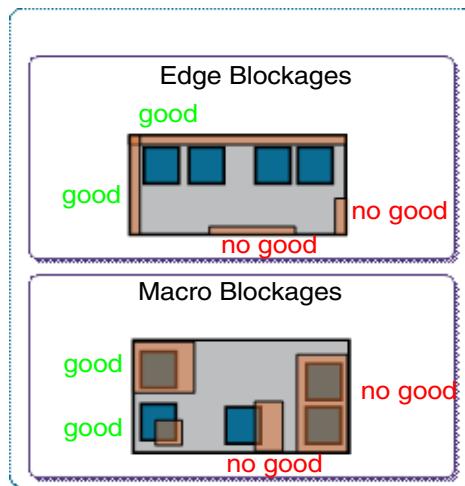
Maximum width – Enter the upper bound of the core width in microns to stop the die search process. By default, no die sizes that result in utilizations less than 0.30 (30 percent) are considered.

Maximum height – Enter the upper bound of the core height in microns to stop the die search process. By default, no die sizes that result in utilizations less than 0.30 (30 percent) are considered.

- Keep blockages – Keeps the placement blockages in the input floorplan. By default, the `estimate_fp_area` command deletes all placement blockages before it starts iterating through design sizes.

Some blockages are recognized as macro blockages or edge blockages, as shown in [Figure 3-5](#).

Figure 3-5 Blockage Types



A blockage that covers any side of a core area is called an “edge blockage.” Edge blockages keep cells and macros from being placed right at the edge of the core area.

Macro blockages are placement blockages assigned to macros. If a placement blockage overlaps a macro such that it covers 70 percent or more of the macro area, the placement blockage is assumed to be associated with the macro. The purpose of a macro blockage is to keep standard cells from being placed too close to the macro.

If a blockage covers two or more macros, it is randomly assigned to one macro.

It is recommended that you create a separate placement blockage for each hard macro rather than one placement blockage over a set of macros.

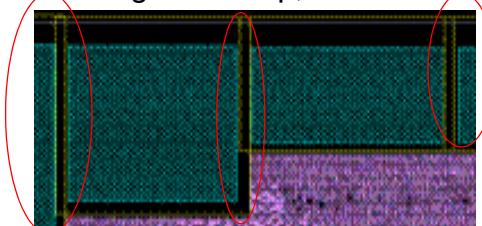
During die size reduction, macro blockages move along with the macros, which may result in blockage overlaps, as shown in [Figure 3-6 on page 3-11](#). Blockage overlaps are allowed. Macro overlaps, however, are not allowed.

Figure 3-6 Blockage Overlaps Allowed; Macro Overlaps Not Allowed

Blockages and macros do not overlap



Blockages overlap; macros do not



Note:

For most designs, the minimum allowable macro spacing can be achieved using the `set_keepout_margin` command and the `sliver_size` parameter value. Creating and maintaining placement blockages to successfully complete estimate area iterations is not necessary. However, the `estimate_fp_area` command honors both soft and hard placement blockages.

If you specify a `sliver_size` parameter value, and the distance between two objects, such as macros, core edges, or blockages, is less than or equal to the `sliver_size` value before running the `estimate_fp_area` command, then this distance is maintained by the `estimate_fp_area` command.

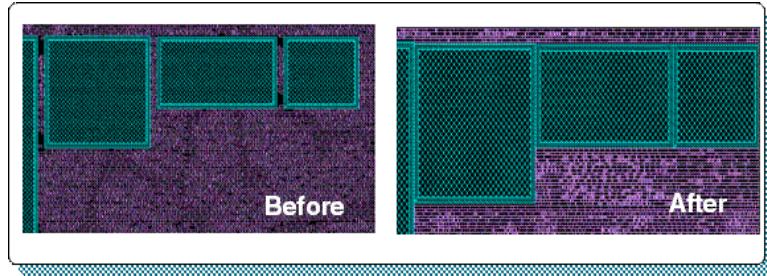
If there are no overlaps between the macro padding, and the distance between the macros is less than or equal to the `sliver_size`, the `estimate_fp_area` command will not reduce the space between the hard macros.

If there are overlaps between the macro padding, and the distance between the macros is less than or equal to the `sliver_size` value, the `estimate_fp_area` command will not reduce the space between the hard macros, and the macro padding will also remain overlapped with the other padding.

If there is no `sliver_size` parameter value or `set_keepout_margin` specified for a fixed macro and one placement blockage is created over a set of multiple hard macros, the macros will abut. When the `estimate_fp_area` command is run, it does not reduce the placement blockage. Instead, it blocks some of the core area, preventing the placement of the standard cells.

If there is a hard macro keepout, macros can be packed until the keepout region abuts, as shown in [Figure 3-7 on page 3-12](#).

Figure 3-7 Set Keepout Region Packing Limit



- Maintain I/O pad alignment – Maintains I/O pad alignment of designs with complex structures, such as rectilinear I/Os, staggered I/Os, multi-ring I/Os, and macros, encroaching into the I/O area, and I/O cells encroaching into the core area.

For multi-ring I/O placement, the `estimate_fp_area` command scales only the spaces that are shared by all I/O rings on the same side of the chip. This helps maintain the desired center or edge alignments between I/O pads across different I/O rings. By default, each I/O ring is scaled separately and proportionally.

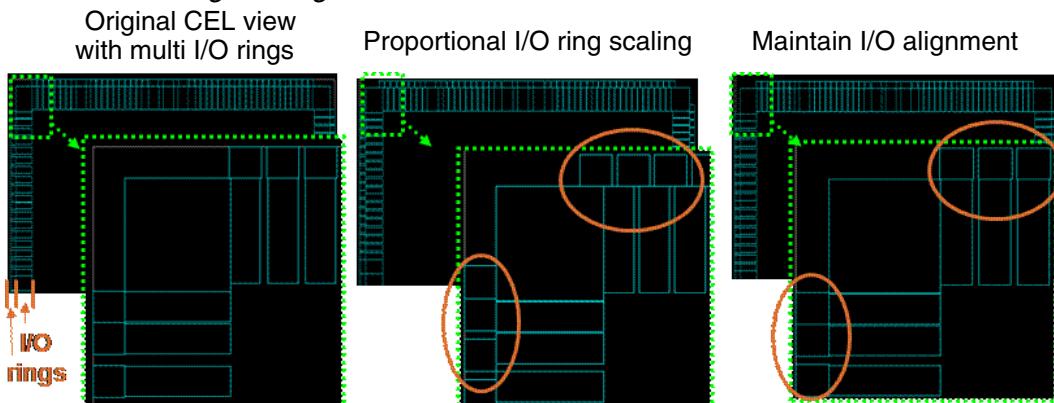
[Figure 3-8](#) shows an example of maintaining I/O pad alignment.

The image on the left is the original CEL view with multi I/O rings. The pads between the I/O rings are aligned.

When the `estimate_fp_area` command is run with the default options, the I/O rings are scaled proportionally. As a result the alignment between the I/O pads in the I/O rings is lost, as shown in the center image.

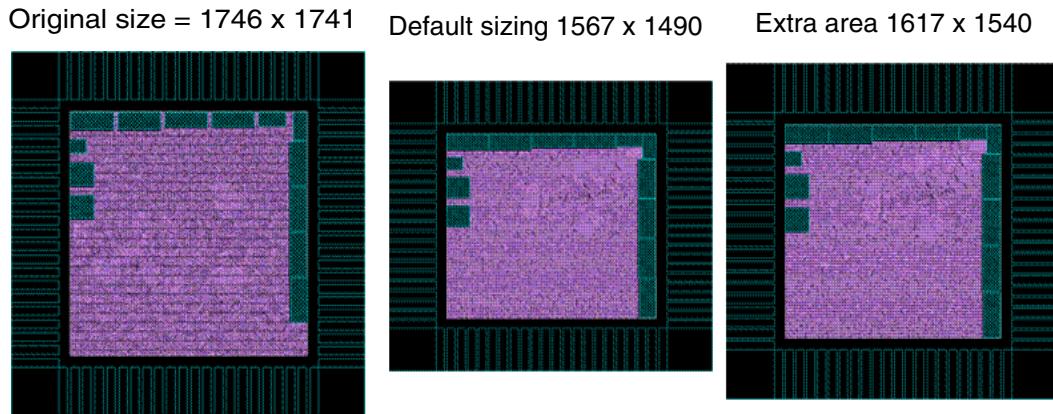
When the `estimate_fp_area` command is run with the “Maintain I/O pad alignment” option selected, the shrinkage occurs by eliminating the unnecessary space in the I/O ring and by maintaining the I/O alignment between the I/O rings, as shown in the image on the right.

Figure 3-8 Maintaining I/O Alignment



- Replace I/O – Replaces all the I/O pads and pins based on the TDF constraints. If a TDF file does not exist, Estimate Area creates one with side and order constraints. By default, the relative I/O placement of the input floorplan is maintained.
- Increase area – Reserves some extra space in the core area for cells to be added later in the flow. This increased space can help accommodate any clock tree synthesis or optimization buffers that might be inserted later during the detailed implementation. Specify the area in square microns. The default is 0. [Figure 3-9](#) shows an example of reserving some space in the core area for future use.

Figure 3-9 Increasing Space in the Core Area



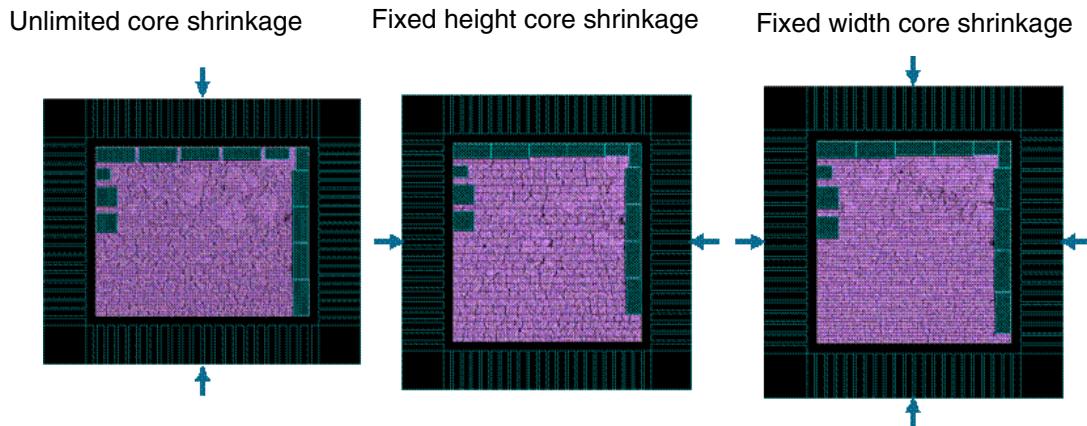
- Core sizing only – Core sizing applies only to full-chip designs. In designs that have complex custom I/O structures, the tool does not modify the I/O structure appropriately. You can shrink the size of the core area only and ignore the I/O pad ring in the input floorplan. Note that if the core size is increased, it will overlap the I/O structure.

This option is provided to support a what-if type of analysis.

By default, the block or chip shrinks with the core and the I/O pad or pin placement is scaled.

[Figure 3-10 on page 3-14](#) shows an example of core sizing within the I/O ring.

Figure 3-10 Core Sizing Only; Core Sizing With Fixed Height and Fixed Width



3. Set the search control options, depending on your requirements.

- Acceptable overflow – This is the main stop criteria. Enter a global route cell (GRC) overflow percentage in the range of 0.0 to 0.25 to determine if the design is routable. When a tested floorplan has the largest achievable global route cell (GRC) acceptable overflow percentage that does not exceed the value you specify, it is saved as the smallest routable floorplan.

Auto Routability – This is the default search control option for estimating routability. No user input is needed, and it runs fast.

Auto routability helps you to better predict a routable floorplan by:

- Predicting the completion rate of detail routing based on global routing results.
- Reducing the pessimism caused by global route cell based local congestion hotspots.
- Predicting the routability more accurately than the global route cell based on analyzing the neighborhoods of individual global route cells.
- Max allowable IR drop value – Allows you to set the acceptable IR drop criteria used by power network synthesis when it creates a power plan during `estimate_fp_area`. This option is not valid unless the “Power and ground net names” field is also used. To use “Power and ground net names” with `estimate_fp_area`, you should first run a trial power network synthesis on the existing floorplan to ensure that all the power network synthesis options are set properly before starting `estimate_fp_area`.

Enter a value in mv units. If you do not specify a value, by default 10 percent of the power supply voltage is used as the target IR drop value.

4. Set the power network control options, depending on your requirements.

- Automatically create script to mimic power network – Creates a power network synthesis script file that is automatically sourced within MinChip to replicate the power structures in the input floorplan for a new die size. The input floorplan should contain a regular power structure with straps or rings. The power structures in the input floorplan are analyzed and the power network synthesis constraints are extracted. Irregular power structures and multivoltage design power structures are not extracted.

MinChip creates the script file `EA_pns_script.tcl` in the current directory.

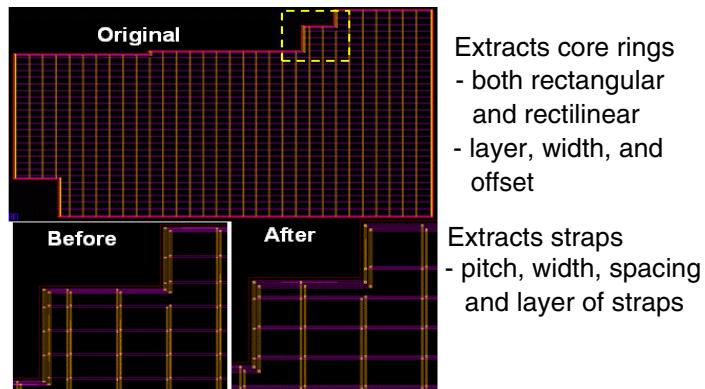
When you select this option, you must also include the power and ground net names.

The extracted power network synthesis script creates core rings with fixed layers and widths, straps with fixed pitch, widths and layers, block rings with fixed layers and widths, and standard cell preroutes.

The die that results should have a power plan that is very similar to that of the input floorplan but scaled to the new die size.

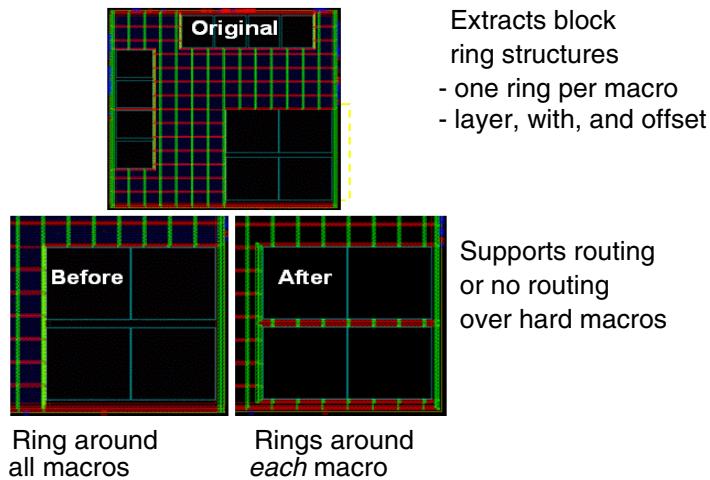
[Figure 3-11](#) shows an example of extracting rectangular and rectilinear core rings, and straps.

Figure 3-11 Extracting the Power Structure: Core Rings and Straps



[Figure 3-12 on page 3-16](#) shows an example of extracting block ring structures. Routing or no routing over hard macros is also supported.

Figure 3-12 Extracting the Power Structure: Block Rings and Routing Over Macros



Minchip honors the virtual pad locations, which are taken from the power network synthesis constraints. The virtual pad file is read and the locations of the virtual pads are automatically scaled to the new die size.

- Power and ground net pairs as {pwr gnd} – Performs power network synthesis using the power network synthesis default settings during each new die size exploration. The default power network synthesis settings form a two-layer mesh on the uppermost layers in the technology that meets the specified voltage drop (IR drop) target. If you do not specify a voltage drop target, the default assumes the target is 10 percent of the supply voltages. If this type of power structure does not accurately represent the power structure used for your design, you should use the “Custom PNS script” option.

The power plan routes provide a more realistic routing overflow estimate because the routing resources used by the power preroutes are taken into account during the global route cell overflow calculations.

You must specify a power and ground net name pair. A comma must be used to separate the names of the power net and ground net. For example,

VDD, VSS

By default, no power net synthesis is run.

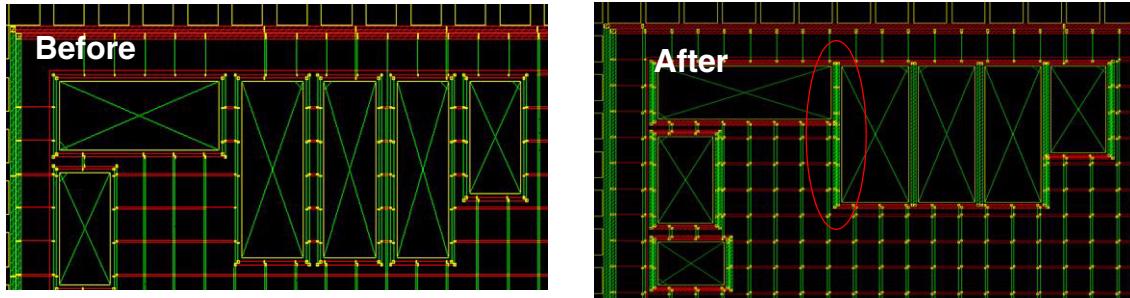
- Custom PNS script – Runs a custom power network synthesis script instead of the default settings used to run power network synthesis.

By default, no power network synthesis is run.

It is important that the script not contain any absolute coordinates as they do not apply to the various design sizes that the `estimate_fp_area` command iterates through.

If you provide a power network synthesis script file with hard macro block rings, but with no `sliver_size` or macro padding, as shown in [Figure 3-13](#), the `estimate_fp_area` command can reduce the die size. The die size is reduced by eliminating the space between the hard macros until the macro block power network synthesis rings are merged with one set instead of two sets.

Figure 3-13 Estimate Area Run With Power Network Synthesis and Fixed Hard Macros



- Preroute script – Runs a user-defined power routing script used to form preroutes to connect the power and ground pins of standard cells (preroute standard cells). During this process, stacked vias are formed to connect the preroutes to the upper layer power routes, if applicable.

It is important that the script not contain any absolute coordinates as they do not apply to the various design sizes that the `estimate_fp_area` command iterates through.

By default, no preroutes are created.

5. Save the result of the die size exploration.

- Save as – Explicitly name the saved CEL view. By default, the tool saves the last routable result as a CEL view named *design_name* EstimateArea.

6. Select OK or Apply.

When the `estimate_fp_area` command is run, it does the following:

- Saves the starting CEL view with the original cell name and starts working on this CEL view.
- Deletes both signal and power routing.
- Maintains the orientation of the fixed macros; it does not rotate the macros.
- First tests the routability of the input CEL view by using the current utilization of this CEL core to set the die size. The `estimate_fp_area` command then attempts to make this die size as small as possible.
- Stops if the design is pad limited. If the design becomes pad limited during estimate area iterations, the command stops and informs you that sizing stopped because the design had become pad limited.

If the original design has PAD min-spacing constraints, a pad limited floorplan will not be reached since PADS cannot be abutted because of the constraints.

Using the `estimate_fp_area` Command and Options

This section describes how to use the `estimate_fp_area` command and its options. The command options are grouped into three categories:

- Search Control
- Floorplan Control
- Power Network Control

The `estimate_fp_area` command automates the placement and routing iterations targeted at finding the smallest, routable die size. It automatically maintains relative placement of fixed hard macros, uses routing congestion estimation technology to assess routing feasibility, and optionally uses power network synthesis for power routing.

The `estimate_fp_area` command uses the following syntax:

```
estimate_fp_area
[-min_height min_height]
[-max_height max_height]
[-min_width min_width]
[-max_width max_width]
[-acceptable_overflow percentage]
[-ir_drop_value target_value]
[-sizing_type fixed_width | fixed_height | fixed_aspect_ratio]
[-core_sizing_only]
{-keep_blockages}
[-increase_area area]
[-replace_io]
[-maintain_iopad_alignment]
[-power_net_names list_of_names]
[-run_preroute_script preroute_script_name]
[-run_pns_script script_name]
[-save_as name]
[-maintain_power_structure]
[-estimate_optimization]
```

Enabling Zroute in MinChip

In addition to the classic global router, the `estimate_fp_area` command supports the Zroute router. To enable Zroute in MinChip, set the following variable before running the `estimate_fp_area` command.

```
set_route_mode_options -zroute true
```

Table 3-1 describes the MinChip Technology search control options.

Table 3-1 MinChip Technology Search Control Options

Search Control Options	Description
<code>-min_height min_height</code>	Specifies the lower bound of the core height, in microns, required to stop the die search process. It stops making a routable core smaller if the total utilization for the core is less than the target value you specify. By default, no die sizes that have utilizations greater than 95 percent are considered.
<code>-max_height max_height</code>	Specifies the upper bound of the core height, in microns, required to stop the die search process. It stops making a routable core larger if the total utilization for the core is greater than the target value you specify. By default, no die sizes that have utilizations greater than 30 percent are considered.
<code>-min_width min_width</code>	Specifies the lower bound of the core width, in microns, required to stop the die search process. It stops making a routable core smaller if the total utilization for the core is less than the target value you specify. By default, no die sizes that have utilizations greater than 95 percent are considered.
<code>-max_width max_width</code>	Specifies the upper bound of the core width, in microns, required to stop the die search process. It stops making a routable core larger if the total utilization for the core is greater than the target value you specify. By default, no die sizes that have utilizations greater than 30 percent are considered.

Table 3-1 MinChip Technology Search Control Options

Search Control Options	Description
<code>-acceptable_overflow_percentage</code>	<p>This is the main stop criterion for the <code>estimate_fp_area</code> command.</p> <p>Auto Routability: This is the default search control option for estimating routability. No user input is needed and it runs quickly. Auto routability helps you to better predict a routable floorplan by:</p> <ul style="list-style-type: none"> Predicting the completion rate of detailed routing based on global routing results Reducing the pessimism caused by global route-based local congestion Predicting the routability more accurately than the global route cell based on neighborhood gcell analysis <p>Acceptable Overflow: If you specify a value with the <code>-acceptable_overflow</code> option, the <code>estimate_fp_area</code> command specifies a global route cell overflow percentage to determine if the design is routable. This is the main stop criteria for the <code>estimate_fp_area</code> command. The Auto Routability feature is disabled.</p> <p>When a tested floorplan has the largest achievable global route cell acceptable overflow percentage without exceeding the value you specify, it is saved as the smallest routable floorplan.</p> <p>Enter the value as a floating point number. The default is 0.018 (1.8%).</p>
<code>-ir_drop_value target_value</code>	<p>Allows you to set the acceptable IR drop criteria used by power network synthesis when it creates a power plan during <code>estimate_fp_area</code>. This option is not valid unless the <code>-power</code> option is used. To use the <code>-power</code> option with <code>estimate_fp_area</code>, it is recommended that you first run a trial power network synthesis on the existing floorplan to ensure that all the power network synthesis options are set properly before starting <code>estimate_fp_area</code>.</p> <p>The value is in mv units. If you do not specify a value, 10 percent of the power supply voltage is used as the target IR drop value.</p>

Table 3-2 describes the MinChip Technology floorplan control options.

Table 3-2 MinChip Technology Floorplan Control Options

Floorplan Control Options	Description
-sizing_type fixed_width fixed_height fixed_aspect_ratio	<p>Defines constraints on the search space of aspect ratios.</p> <p>Select <code>fixed_width</code> if you want to preserve the original design width and perform sizing in the y-direction only.</p> <p>Select <code>fixed_height</code> if you want to preserve the original design height and perform sizing in the x-direction only.</p> <p>Select <code>fixed_aspect_ratio</code> if you want the <code>estimate_fp_area</code> command to honor the aspect ratio of the original design size and to stop shrinking the width or height of the die when the width or height becomes pad or macro-limited.</p> <p>By default, the <code>estimate_fp_area</code> command ignores the aspect ratio and continues shrinking the width or height of the die until the width or height becomes pad or macro-limited.</p>
-increase_area area	<p>By default, the <code>estimate_fp_area</code> command finds the smallest, routable die size. Specify this option to reserve some extra space for cells to be added later in the flow.</p> <p>This increased space can help accommodate any clock tree synthesis or optimization buffers that might be inserted during the detailed implementation.</p>

Table 3-2 MinChip Technology Floorplan Control Options

Floorplan Control Options	Description
-maintain_iopad_alignment	<p>Aligns the I/O pads in different rings when multi-ring I/O structures are present. For multi-ring I/O placement, the <code>estimate_fp_area</code> command scales only the spaces that are shared by all I/O rings on the same side of the chip. This helps maintain the desired center or edge alignments between I/O pads across different I/O rings.</p> <p>The <code>estimate_fp_area</code> command scales only the spaces that are shared by all I/O rings on the same side of the chip. This helps maintain the desired center or edge alignments between I/O pads across different I/O rings.</p> <p>It also handles designs with complex I/O structures, such as, rectilinear I/Os, staggered I/Os, multi-ring I/Os, and macros, encroaching into the I/O area, and I/O cells encroaching into the core area.</p> <p>Shrinkage occurs by eliminating the unnecessary space in the I/O ring and by maintaining the I/O alignment between the I/O rings.</p> <p>By default, each I/O ring is scaled separately and proportionally.</p>
-keep_blockages	<p>Keeps the placement blockages in the input floorplan. The blockages are not deleted.</p> <p>By default, the <code>estimate_fp_area</code> command deletes all placement blockages in the input floorplan before it starts iterating through design sizes.</p>
-replace_io	<p>Replaces all I/O pads and pins based on Top Design Format (TDF) constraints. If no TDF file exists, one is created with side and order constraints.</p> <p>By default, the <code>estimate_fp_area</code> command maintains the relative I/O placement of the input floorplan.</p>

Table 3-2 MinChip Technology Floorplan Control Options

Floorplan Control Options	Description
-core_sizing_only	<p>Applies only to full-chip designs. It modifies the size of the core area and ignores the I/O pad ring in the input floorplan. In designs with complex custom I/O structures, the <code>estimate_fp_area</code> command may not modify the I/O structure appropriately. This option can be used to ignore the I/O structure and modify the size of the core only. Note that if the core size is increased, it will overlap the I/O structure.</p> <p>By default, the block or chip shrinks with the core and the I/O pad or pin placement is scaled.</p>
-estimate_optimization	<p>Estimates the number of buffers added during optimization and reserves the area needed for buffer insertion. Therefore, the final die size after running the <code>estimate_fp_area</code> command will include the area needed for the current netlist plus some additional area reserved for optimization. MinChip will output the amount of area reserved for buffer insertion.</p>

Table 3-3 describes the MinChip Technology power network control options.

Table 3-3 MinChip Technology Power Network Control Options

Power Network Control Options	Description
-power_net_names <i>list_of_names</i>	<p>Performs power network synthesis during each new die size exploration in <code>estimate_fp_area</code>. The power plan routes provide a more realistic routing overflow estimate because the routing resources used by the power preroutes are taken into account during the global route cell overflow calculations.</p> <p>You must specify a power and ground net name pair. A comma must be used to separate the names of the power net and ground net. For example,</p> <p style="padding-left: 20px;">-power VDD, VSS</p>

Table 3-3 MinChip Technology Power Network Control Options

Power Network Control Options	Description
<code>-run_pns_script <i>script_name</i></code>	<p>Causes the <code>estimate_fp_area</code> command to run a user-defined power network synthesis script instead of the default settings to run power network synthesis.</p> <p>By default, power network synthesis is run unless you use either the <code>-run_pns_script</code> or the <code>-power_nets</code> option.</p> <p>The script should not contain any absolute coordinates as these absolute coordinates do not apply to the various design sizes that the <code>estimate_fp_area</code> command iterates through.</p>
<code>-run_pre_route_post_pns <i>preroute_script_name</i></code>	<p>Causes the <code>estimate_fp_area</code> command to run a user-defined power routing script to form preroutes to connect the power and ground pins of standard cells. During this process, stacked vias are formed to connect the preroutes to the upper layer power routes, if applicable.</p> <p>The script should not contain any absolute coordinates as these absolute coordinates do not apply to the various design sizes that the <code>estimate_fp_area</code> command iterates through.</p>
<code>-maintain_power_structure</code>	<p>Creates a power network synthesis script file that is automatically sourced within MinChip to replicate the power structures in the input floorplan for a new die size.</p> <p>The resultant die should have a power plan that is very similar to that of the input floorplan but scaled to the new die size.</p>
<code>-save_as_cell <i>cell_name</i></code>	<p>Use this option to name the saved CEL explicitly.</p> <p>By default, the <code>estimate_fp_area</code> command saves the last routable result as a CEL named <i>design_name</i> EstimateArea.</p>

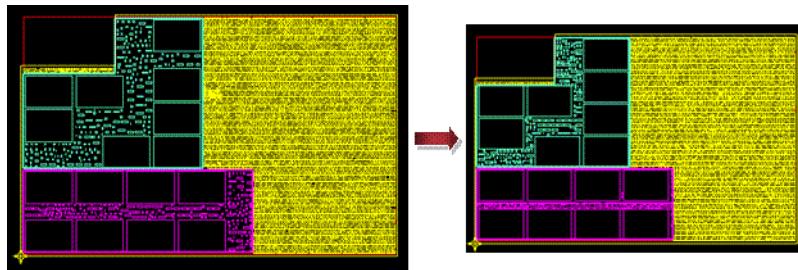
Using Multivoltage Designs in MinChip Technology

MinChip technology supports designs with voltage areas. When the `estimate_fp_area` command iterates through different design size estimates to find the smallest die or block size that meets the routability target, it automatically shrinks rectangular and rectilinear voltage area boundaries by a similar percentage, as shown in [Figure 3-14](#).

The `estimate_fp_area` command shrinks the voltage area boundary by the same amount as the top level block or cell. The relative macro placement of the top level hard macros as well as the macros inside the voltage areas is maintained.

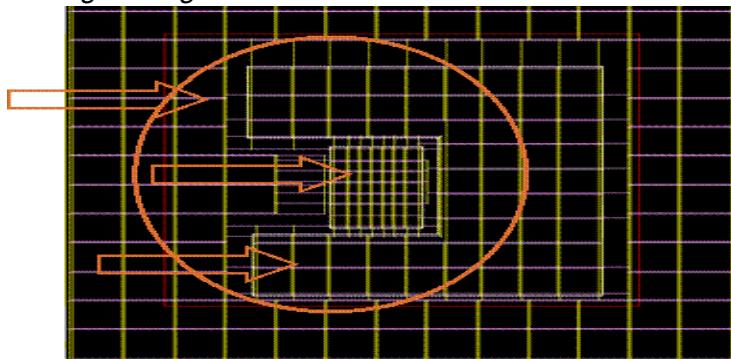
If a voltage area cannot be placed correctly because the utilization bounds are too high, the `estimate_fp_area` command stops shrinking the design. It also ensures that the aspect ratio (shape) voltage area changes in accordance with any changes to the aspect ratio of the top-level block or cell.

Figure 3-14 Voltage Area Sizing



Based on the power budget requirements, each voltage area in the core area can have different power structures, as shown in [Figure 3-15](#). Therefore, it is recommended that you use a power network synthesis script because the default run may not generate different power structures for different voltage areas.

Figure 3-15 Multivoltage Designs/Power Structure



4

Handling Black Boxes

This chapter describes the various operations you can perform with black box modules and cells in IC Compiler, using a virtual flat approach to creating the floorplan.

Milkyway defines a black box as any instance in the netlist that does not have either a corresponding leaf cell in a reference library or a netlist module defined in terms of a leaf cell. Therefore, a black box can have an empty module represented in the netlist where its ports and their directions are defined. If the black box instance in the netlist does not have a corresponding module, the ports of the black box are defined in the Milkyway database with inout directions.

Black boxes can be represented in the physical design as either soft or hard macros. A black box macro has a fixed height and width. A black box soft macro sized by area and utilization can be shaped to best fit the floorplan.

This chapter includes the following sections:

- [Reading Netlists With Black Boxes](#)
- [Creating CEL Views for Black Boxes](#)
- [Sizing Black Boxes by Gate Equivalence](#)
- [Estimating Black Box Sizes](#)
- [Managing the Attributes for a Black Box](#)
- [Flattening Existing Black Box Instances](#)

- [Creating Quick Timing Models for Black Boxes](#)

Reading Netlists With Black Boxes

To read a Verilog netlist, use the following command:

```
read_verilog verilog_file
```

For a timing-driven flow, you must create a FRAM view for the black box after performing black box pin assignment using the non-timing driven flow. After you generate the FRAM view, set it as a reference library and continue the flow from timing-driven placement.

When IC Compiler reads the Verilog netlist, the following module types are detected as black boxes:

- Empty

This is the module type that you are most likely to want to keep as a black box. Empty-type modules have a port list and usually have input and output ports designated in the module definition as well.

Here is an example of an empty module:

```
module Tuner (a,b,z);
  input a,b;
  output z;
endmodule
```

- Missing

This module type can be an intended black box or it can indicate a missing reference library link. If one or more reference libraries are not linked to the top design library, literally thousands of these missing black box modules might be detected.

To limit the amount of time the tool might spend finding such modules, you can set the number of black boxes that are detected and listed in the text box at the bottom of the Import Black Boxes dialog box. (Choose Floorplan > Blackboxes > Import Blackboxes) The default limit is 100. You can increase this number in the Max Num field if you have a design with more than 100 expected real black boxes.

If you see many missing-type black boxes showing up in the importable black boxes list and you were not expecting to see any or were expecting to see only few, check the listed module names. If you discover that these missing-type modules are really leaf cell references, add the missing reference library link. This allows you to avoid spending time creating CEL views for instances that are really leaf cell references.

- Feedthrough

If a module has assign statements in which all the ports of the module are found on either the left or right side of an assign statement in the module, it is designated as a feedthrough type. Feedthrough is a subset of the Design Flow (DF) type.

Here is an example of a feedthrough module in Verilog:

```
module bus_function (In,Out);
    input [7:0] In;
    output [7:0] Out;

    assign Out=In;
endmodule
```

- Data flow (DF)

If any port on the module is not covered in an assign statement, it is designated as a DF type. (The DF signifies “data flow,” module from Verilog nomenclature.)

Here is an example of a DF module in Verilog:

```
module ip_bus_tieoff
(Z);
    output [7:0] Z;
    output X;

    assign Z = 8'b0;
endmodule
```

- Tie-off

If a module has assign statements with only output ports and those output ports are connected to either 1'b0 or 1'b1 within that module, it is designated as a tie-off type. Tie-off is a subset of the DF type.

Here is an example of a tie-off module in a netlist:

```
module ip_bus_tieoff (Z);
    output[7:0]Z;

    assign Z=8'b0;
endmodule
```

Creating CEL Views for Black Boxes

You can interactively import one or more black boxes from the hierarchy preservation information (logical view) into a physical CEL view to allow prototyping of the black box module types.

To create CEL views for black boxes, choose Floorplan > Blackboxes > Import Blackboxes.

The Import Black Boxes dialog box appears.

Alternatively, you can use the `import_fp_black_boxes` command.

- You can select from a list of instance names of the black boxes in the Hierarchy Preservation information that can be imported into the CEL views. Clicking the Select All button creates CEL views for every black box listed.
- Estimate size – If you select this option, you can enter a “Side length” by specifying the length of one side of the created black box. The cell boundary is considered to be the absolute sizing standard.

Note:

By default, each initial black box is imported and sized to fit a 300 x 300 square black box boundary (300 tech file distance units on each of its four sides).

- Enclosed standard cells – If you select this option, you can enter an integer, which is a count of equivalent base gate cells used to calculate the area of the automatically sized black box boundary.
- Utilization – Enter a floating-point value to define the utilization factor used for black box creation. The default is 0.7.
- Maximum number of black boxes to import –The maximum number of black boxes that can be imported at any one time is 100. If the specified number of black boxes to import exceeds this number (the default is 100), the import is canceled. This is to prevent you from accidentally selecting all black boxes in a script when you might have forgotten to specify one or more reference libraries, which, in turn, can result in importing hundreds of black boxes.

The following are examples of how to import each of the five defined black box types. By default, all empty- and missing-type black boxes are selected for import.

```
import_fp_black_box [get_cells -hier -filter  
"is_logical_black_box==true && black_box_type==Empty"]  
  
import_fp_black_box [get_cells -hier -filter  
"is_logical_black_box==true && black_box_type==Missing"]  
  
import_fp_black_box [get_cells -hier -filter  
"is_logical_black_box==true && black_box_type==Feedthru"]  
  
import_fp_black_box [get_cells -hier -filter  
"is_logical_black_box==true && black_box_type==Tie-Off"]  
  
import_fp_black_box [get_cells -hier -filter  
"is_logical_black_box==true && black_box_type==DF"]
```

Sizing Black Boxes by Gate Equivalence

You can use gate equivalence calculations to estimate the size of black boxes. To do this, you must run the `set_fp_base_gate` command, so that there is a user-defined base gate size. The gate equivalent of an empty macro cell is the number of gates, such as 2-input NAND gates, needed to occupy the same area as the actual cells in the hard macro, considering its current utilization. This gate equivalence base unit can be set as a gate from the library or an area.

- Select the `-cell` option to use a library standard (std) cell for the gate equivalence area base reference. The industry standard cell for this setting is a 2-input NAND gate.

For example,

```
set_fp_base_gate -cell nd02
```

- Select the `-area` option to set a base area for gate equivalence calculations.

For example,

```
set_fp_base_gate -area 24
```

Estimating Black Box Sizes

You can estimate the size of a black box so that it can be treated as a normal soft macro in the rest of the design flow. The estimated size can be the sum of the sizes of the estimated enclosed leaf cells, soft macros, or hard macros the black box represents.

To estimate a black box as a soft or hard macro, choose Floorplan > Blackboxes > Estimate Blackboxes.

The Estimate Black Boxes dialog box appears.

Alternatively, you can use the `estimate_fp_black_boxes` command.

- Black boxes – Specify a list of black boxes to estimate.
- Estimate as soft macro – If you select this option (the default), you can set the estimated width and the height of the soft macro representing the black box.

A black box that is estimated as a soft macro will be a CEL view in the layout window and will have a thick boundary outline, just like any other soft macro.

- Soft macro polygon – If you select this option, you can reshape the black box into a rectilinear black box. It modifies the black box's cell boundary so that it is defined by the list of points of the polygon you want for the polygon area. The points are interpreted as absolute coordinates in microns. The format is `{ {x1 y1} {x2 y2} ... {xn yn} }`. The utilization is set to a value of 1.0.

The default is off.

- Enclosed standard cells – If you select this option, you can set a gate equivalence value (gate count) for the standard cells in the black box and you can change the utilization factor: the default is 0.7. The gate equivalence area is divided by the utilization factor you enter, to determine a final black box boundary estimate.

Note:

You can set a value for standard cells in the black box and supply a utilization factor less than 1 while adding in some area for hard macros into the black box size estimate.

- Estimate by enclosed hard macros – You can include a list of real leaf cell macros to be part of the sizing definition for a black box by clicking in the text box and entering a comma-separated list of hard macro names.

A black box estimated as a hard macro will be a FRAM view in the layout window. It will have a thin boundary outline, just like any other hard macro.

The format is:

name count

where *name* is the name of the hard macro and *count* is an optional number of instances in parentheses, appended to the name. For example,

ALU MUX (3)

Note:

If only one particular hard macro is to be included for sizing, you can omit the number in parentheses for that macro. The list of macros used for sizing a black box is used only for the purpose of adding area to a black box. The shapes of the hard macros are not considered, so it is recommended that you enter the width and height of the hard macro if you need to influence the shape of the black box with the shape of an included hard macro.

- Mark as fixed shape after estimation – Select this option to prevent the placer from resizing the black box shape after estimation.
- Reset shape to rectangular if currently rectilinear – Select this option to reset a black box to a rectangular shape if it is currently rectilinear.

Managing the Attributes for a Black Box

Only the `estimate_fp_black_boxes` command automatically changes the `unestimated` attribute for a black box. If a black box is user-sized by manual editing, the `estimated` attribute does not change. You can manage this attribute to monitor which black boxes have been sized by using the following two commands:

- The `set_fp_black_boxes_estimated` command specifies that the black box object types have been estimated and are therefore valid for floorplanning.
- The `set_fp_black_boxes_unestimated` command sets the specified black box object types as unestimated, so that they are flagged and sized correctly before floorplanning.

If you run the following command on the current black boxes, the value returned will be “unestimated” because all the current black boxes have not been user-estimated yet.

```
icc_shell> get_attribute BB_instance_name sm_estimation_mode
```

Flattening Existing Black Box Instances

When you create a CEL view for a black box module, the layout window contains an associated instantiation of that black box. If you later decide that the black box instantiation is not needed in the layout, use the `flatten_fp_black_boxes` command to remove the physical box instance from the top design in the layout window and restore it to the hierarchy preservation (logical view). The CEL view for the black box is retained in the library, but it will not have any effect on the top design where it was flattened.

The flattened black box still resides in the Verilog netlist output as a module, just like it did in the original netlist.

Creating Quick Timing Models for Black Boxes

A quick timing model is an approximate timing model that is useful early in the design cycle to describe the rough initial timing of a block that has no netlist—for example, a black box. A black box in IC Compiler design planning is defined as any instance in the Verilog netlist that does not have either a corresponding leaf cell in a reference library or a netlist module defined in terms of leaf cells. Therefore a black box might have an empty module represented in the netlist where its ports and their directions are defined. If the black box instance in the netlist does not have a corresponding module, the ports of the black box are defined in the Milkyway database as inout ports. Such a black box is called a “missing” black box in IC Compiler design planning.

To create a quick timing model for a black box, you use PrimeTime commands to specify the model ports, the setup and hold constraints on the inputs, the clock-to-output path delays, and the input-to-output path delays. You can also specify the loads on input ports and the drive strength of output ports.

This section includes the following topics:

- [Using the Quick Timing Model Command Flow for Black Boxes](#)
- [Creating a Quick Timing Model for Black Boxes](#)

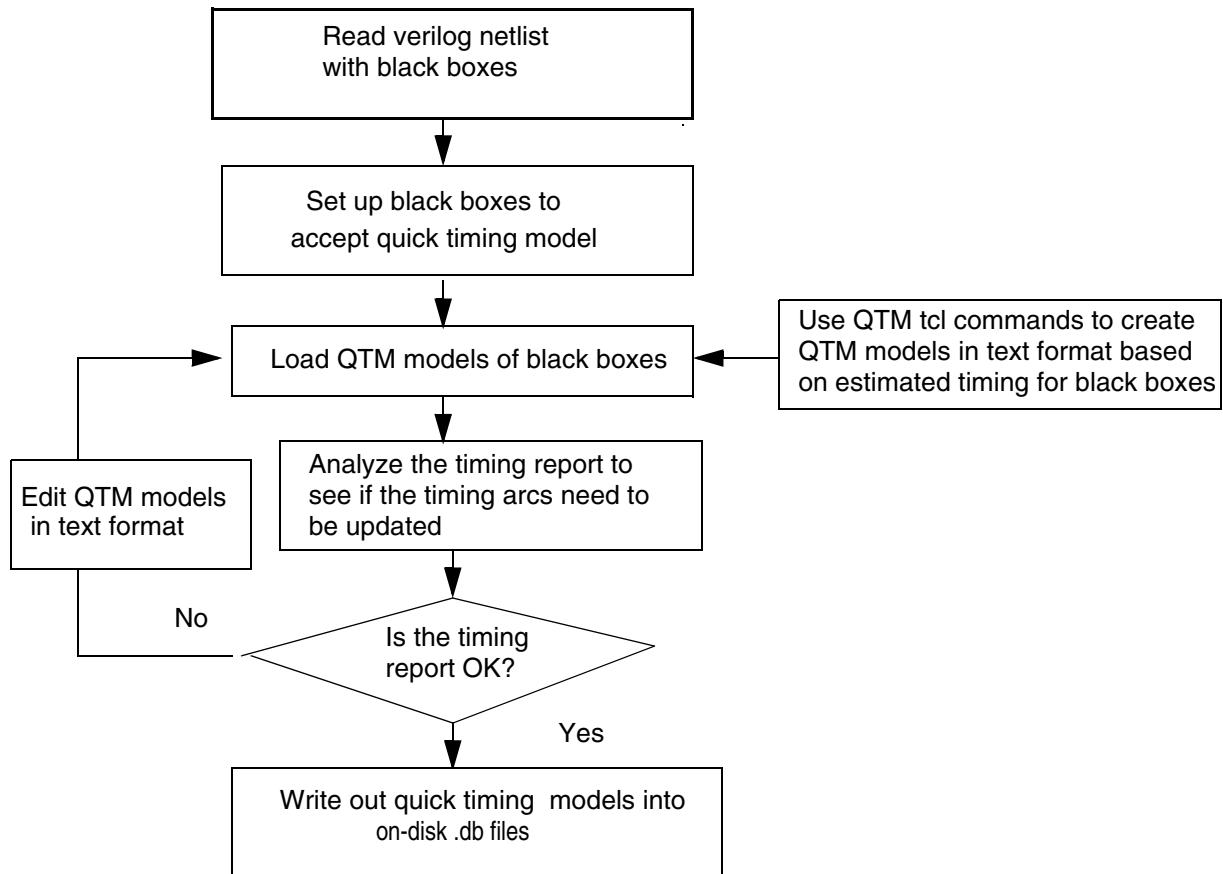
Using the Quick Timing Model Command Flow for Black Boxes

The quick timing model flow provides a way to create timing arcs for the black box macro and to update the timing arcs. You can use the quick timing model flow in IC Compiler to define the quick timing model for black boxes in your design. First, you create a quick timing model file for all the black boxes in the design. Library Compiler uses the information in the quick timing model files to create corresponding Synopsys .lib files and then .db files with the appropriate timing arcs. The Synopsys .db files are then combined into one .db file. The combined Synopsys .db file is attached to the Logic Modeling view of the current design library. You use the `report_timing` command generate a report of the timing paths that pass through ports of the black boxes.

You can continue to update (edit) the black box timing arcs by performing “what-if timing budgeting” until the full top-level timing analysis results are clean.

[Figure 4-1 on page 4-9](#) shows the quick timing model flow.

Figure 4-1 Quick Timing Model Flow



Prerequisites for Running the Quick Timing Model Flow

Before you create quick timing models for black boxes, make sure you have done the following:

- Read in the Verilog netlist containing black box instances by using the `read_verilog` command.
- Provided an estimated shape for the black box by using the `estimate_fp_black_boxes` command.

A black box has only a module name and boundary ports and does not represent any logical information. If an equivalent gate count is to be used as the parameter for estimating the black box size, you must specify the following inputs when you estimate black boxes:

- The standard cell that the IC Compiler tool uses for gate equivalence calculations through the `set_fp_base_gate` command.

- An estimate of the number of standard cells that the black box contains
- The utilization of the physical model

You can also manually specify the estimated height and width of the black box CEL view boundary.

The black boxes can be represented in the physical design as either soft or hard macros.

- Run virtual flat placement to place the black box macro in the core area. The `create_fp_placement` command does not change the shape of the black boxes it places. You can also use this command after initial virtual flat placement to further refine the placement and shaping of black boxes in the core. Some manual editing might be necessary to finalize the floorplan after these two placement steps.
- Assign all the pins on the black box macros to create physical pins (`place_fp_pins` command).

The black box must be physically at the top level of the currently open CEL view during pin assignment for the pin assignment engine to be able to create any pins on the black box CEL view.

Creating a Quick Timing Model for Black Boxes

The creation of a quick timing model starts with the `create_qtm_model` command and ends with the `save_qtm_model` command. All the commands between `create_qtm_model` and `save_qtm_model` define one quick timing model.

To create a quick timing model for black boxes,

1. Specify that a new model is being created.

```
create_qtm_model qtm_bb
```

2. Specify the technology library to use for creating the quick timing model. Information includes the name of the technology library, the maximum transition time, the maximum capacitance, and the wire load information.

```
set_qtm_technology -lib library_name
```

3. Define load and drive types.

The `create_qtm_load_type` command defines load types that are used to specify the net capacitance of input ports. A library cell constitutes a load type.

The `create_qtm_drive_type` command sets the drive type for each output port; the drive type can be any library cell.

For example,

```
create_qtm_load_type LOAD -lib_cell $flop_name
```

```
create_qtm_drive_type DRIVE -lib_cell $bufname
```

4. Specify global setup and hold characteristics.

```
set_qtm_global_parameter -param setup -lib_cell \
                         $flopname -clock $CLK_pin -pin $D_pin
set_qtm_global_parameter -param hold -value 0
set_qtm_global_parameter -param clk_to_output -lib_cell \
                         $flopname -clock $CLK_pin -pin Q_pin
```

5. Create clock ports.

```
create_qtm_port -type clock $port
```

6. Create input ports, output ports, and inout ports.

```
create_qtm_port -type input $port
create_qtm_port -type output $port
create_qtm_port -type inout $port
```

7. For all input and inout ports, define setup and hold arcs.

```
create_qtm_constraint_arc -setup -edge rise -name \
                           setup_${clock}_${port} -from $clock -to \
                           $port -value 0

create_qtm_constraint_arc -hold -edge rise -name \
                           hold_${clock}_${port} -from $clock -to $port -value 0
```

8. For all output and inout ports, specify an output delay and drive type.

```
set_qtm_port_drive $port -type DRIVE
create_qtm_delay_arc -name delay_${clock}_${port} -from \
                      $clock -to $port -value 0 -edge rise
```

9. (Optional) Generate a report that shows the defined parameters, the ports, and the timing arcs in the quick timing model.

```
report_qtm_model
```

10. Save the quick timing model.

```
save_qtm_model
```

11. Write out the .db file for the quick timing model.

```
write_qtm_model -out_dir dp_for_qtm
```

12. Add the .db file to the “link library” variable.

```
lappend link_library "dp_for_qtm/qtm_bb.db"
```

13. Analyze the timing report to get a report of the timing arcs (paths) that pass through the black boxes.

```
report_timing
```

Quick Timing Model Example of a Black Box

[Figure 4-2](#) shows a quick timing model representation of a black box. The constraint arcs appear as dashed lines and delay arcs appear as solid lines. Port CLK is a clock port, ports A and B are input ports, and ports X and Y are output ports.

The arcs are

SetupA

Constrains port A; the constraining port is clock CLK.

HoldA

Constrains port A; the constraining port is clock CLK.

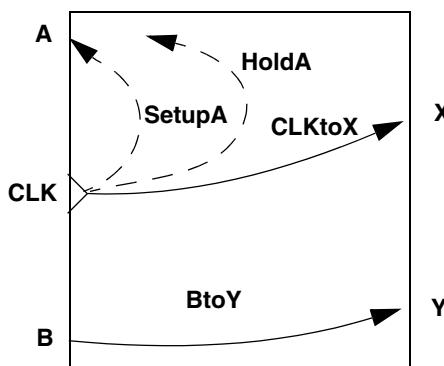
CLKtoX

Sequential delay arc from CLK to X.

BtoY

Combinational delay arc from B to Y.

Figure 4-2 Quick Timing Model Representation of a Black Box



The following tasks were used for specifying the quick timing model in [Figure 4-2](#).

- Create the quick timing model.
- Set the global parameters.
- Specify the model information.
- Print a report of the quick timing model and check the results
- Save the model.

5

Performing an Initial Virtual Flat Placement

The initial virtual flat placement is very fast and is optimized for wire length, congestion, and timing.

This chapter includes the following sections:

- [Evaluating Initial Hard Macro Placement](#)
- [Specifying Hard Macro Placement Constraints](#)
- [Placing Hard Macros and Standard Cells](#)
- [Supporting Relative Placement Groups in Initial Virtual Flat Placement](#)
- [Placing Multiply Instantiated Modules](#)
- [Generating a QoR Placement Report](#)
- [Performing Floorplan Editing](#)
- [Enabling the Feasibility Flow for Optimizing Dirty Constraint Data](#)

Evaluating Initial Hard Macro Placement

No straightforward criteria exist for evaluating the initial hard macro placement. Measuring the quality of results (QoR) of the hard macro placement can be very subjective and often depends on practical design experience. You should observe some basic rules when measuring QoR, such as pushing hard macros to the core boundary and aligning similar hard macros. However, the critical measurements that are used to evaluate the placement results are timing and routability.

QoR can be measured by the following criteria:

- Routability

You can measure the routability of your design by analyzing the routing congestion produced by the global router. The data used to analyze congestion consists of a textual report and visual heat maps that show where the routing congestion “hot spots” exist in the design. Highly congested designs are generally not routable. Hard macros tend to create congestion around their edges and corners. You should analyze hard macro placements for routability early in the design cycle.

- Timing

Timing calculations can use the placement information to better estimate interconnect loading. The interconnect timing calculations should take into account detours in routes caused by the presence of hard macros. Good placement will minimize timing violations.

- Wire length

Smaller values of total wire length are a good indicator of placement quality. The total wire length is output in the placement log file. You should record and monitor this number when assessing multiple placement solutions.

Another aspect of wire length is localized to hard macros. By looking at the signal (flyline) connections of a hard macro, you can quickly determine whether the hard macro is placed in an optimal location. A hard macro placed in the lower-left corner of the core area that is connected to logic in the upper-right corner of the die indicates a poor location for the hard macro.

- Data flow

If you know the logic and intended operation of the design, you can assess the data flow by using the hierarchical browser at any stage in the design flow to observe where logical modules and hard macros are physically placed. Using this technique can help you make sensible placement decisions.

- Standard cell placement areas

You can visually assess the placement of macros and standard cells. Small areas surrounded by hard macros usually cause congestion hot spots. Unless the connections to and from standard cells in these areas are completely localized, it is difficult to complete the connections from within these areas to the objects outside these areas. Generally, a contiguous standard cell placement area without bottlenecks is desirable.

After passing these QoR measurements, the placement result qualifies as a good starting point for further manual tuning.

Specifying Hard Macro Placement Constraints

The following sections describe the different methods you can use to control the preplacement of hard macros and improve the QoR of the hard macro placement.

- [Creating a User-Defined Array of Hard Macros](#)
- [Setting Floorplan Placement Constraints On Macro Cells](#)
- [Placing a Macro Cell Relative to an Anchor Object](#)
- [Using a Virtual Flat Placement Strategy](#)
- [Creating Macro Blockages for Hard Macros](#)
- [Padding the Hard Macros](#)

Creating a User-Defined Array of Hard Macros

You can create a user-defined array of hard macro cells that constrain hard macros to form a one-dimensional or two-dimensional array. The macro cells are placed in the array according to the constraints specified for the array.

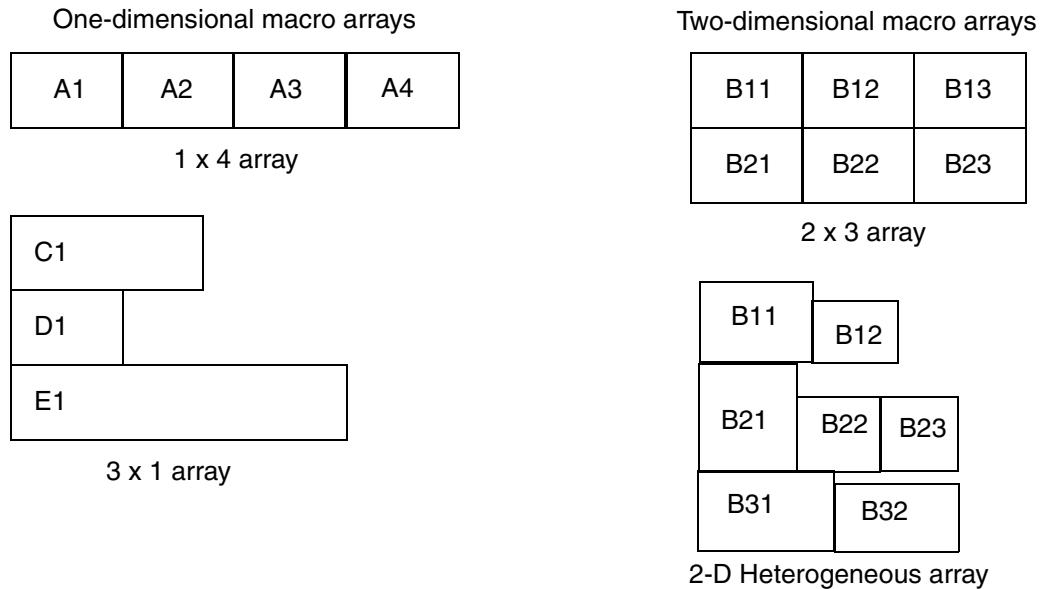
When you create a user-defined array of hard macros, keep the following points in mind:

- The core area must be large enough to accommodate large macro arrays.
- You must specify the order of the macros for the array.
- You can group macros to form a one-dimensional or a two-dimensional array.
- Align one-dimensional and two-dimensional heterogeneous macro arrays by using the “Align edges” option.
- Hard keepout areas are automatically created between the elements in the array.

If there are conflicting constraints on the macros, error messages are printed during placement. However, the error messages do not print details about the conflicts.

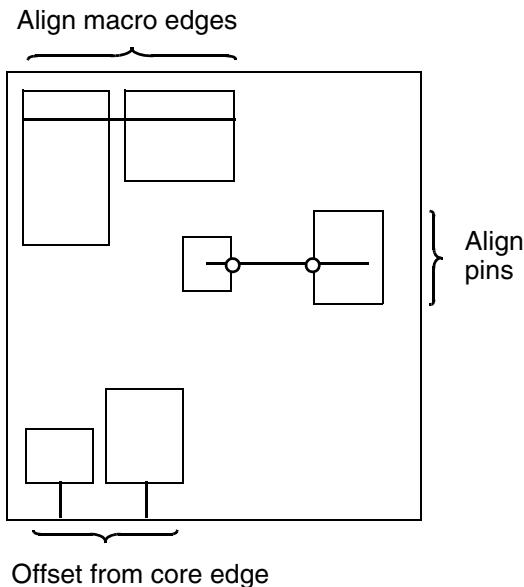
[Figure 5-1](#) shows one-dimensional and two-dimensional macro arrays.

Figure 5-1 One-Dimensional and Two-Dimensional Macro Arrays



[Figure 5-2](#) shows macro alignment to other macros, pins, and edges.

Figure 5-2 Alignment to Other Macros, Pins, and Edges



To create user-defined array of hard macro cells,

1. Choose Placement > Macro Constraints.

The Macro Constraints dialog box appears.

Alternatively, you can use the `set_fp_macro_array` command.

Each constraint is displayed with a unique name, type, and list of macros or macro arrays.

You can sort the list by name or type by clicking on the column header. Selected macro constraints are annotated graphically in the layout window. The original macro cells are also highlighted in the layout window.

2. To create a new macro array constraint, select the Add button on the Macro Constraints dialog box.

From the pop-up menu choose Macro Array.

The Add Macro Array dialog box appears.

3. Set the options, depending on your requirements.

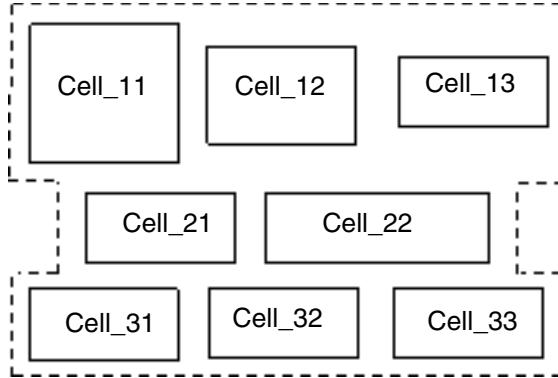
- Macro array name – Specify the macro array name. This name is required.
- Macro cells – You can set the order for the macro cells that will compose the array by using lists and sublists. The macro array template shows how selected macro cells are arranged into an array.

Specify each row of the array as a single list. Each sublist represents a row in the array. An array starts from the topmost row, and a row starts from the left. Based on the list, IC Compiler automatically determines whether to create a one-dimensional or two-dimensional array.

- Rectilinear and Vertical – Select the Rectilinear option to create a two-dimensional array with a rectilinear outline. You can also use the Vertical option with the Rectilinear option to create a vertical row structure for the two-dimensional array. By default, a rectilinear array has a horizontal row structure.

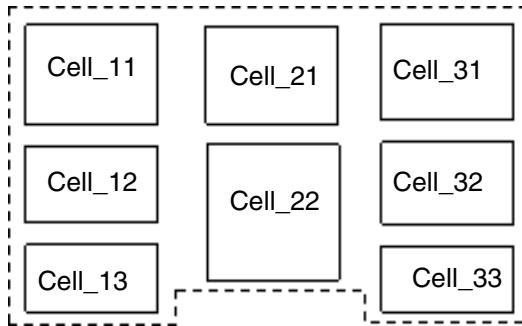
[Figure 5-3 on page 5-6](#) shows an example of a rectilinear horizontal 2-D heterogeneous array.

Figure 5-3 Rectilinear Horizontal 2-D Heterogeneous Array



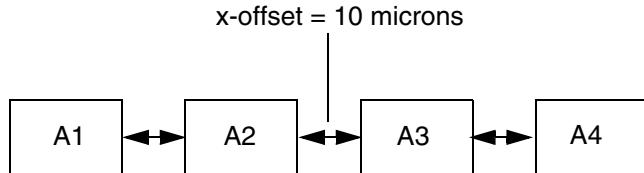
[Figure 5-4](#) shows an example of a rectilinear vertical 2-D heterogeneous array.

Figure 5-4 Rectilinear Vertical 2-D Heterogeneous Array

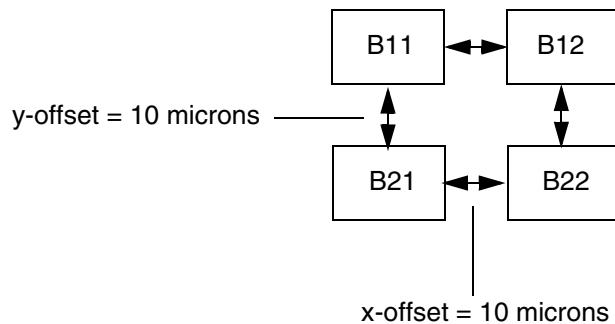


- Use Keepout Margin – When arraying two adjacent macros, you can apply the Use Keepout Margin option to leave the smallest possible spacing between the rows or columns of the two macro arrays. The spacing is greater than or equal to the keepout margins on the two macros.
This option is mutually exclusive with the X offset and Y offset options.
- Specified offset – You can specify the distance between two adjacent cells in a macro array in the x-direction, y-direction, or both directions by using the X offset and Y offset options.

[Figure 5-5 on page 5-7](#) shows the result of applying an x-offset.

Figure 5-5 Result of Applying an X-Offset

[Figure 5-6](#) shows the result of applying both x- and y-offsets.

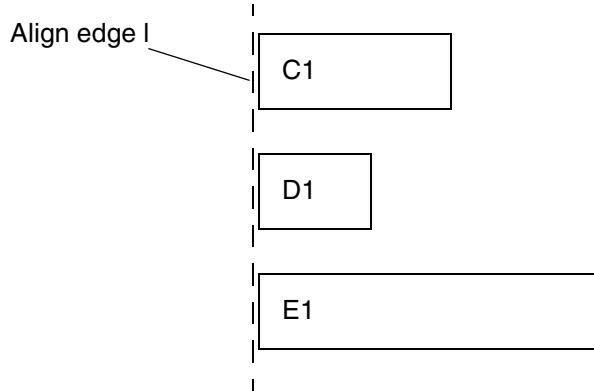
Figure 5-6 Result of Applying X- and Y-Offsets

- Align edges – You can align one row of the macro cells of a heterogeneous one-dimensional macro array along an edge that you specify by selecting this option. The edge can be top, bottom, left, right, or center. The default is bottom.

If you use this option for an array type other than a heterogeneous one-dimensional array, a warning appears and the option is ignored.

If you specify a one-dimensional heterogeneous array and do not use this option, the default is to align the centers.

[Figure 5-7](#) shows the result of applying left alignment.

Figure 5-7 Result of Applying Left Alignment

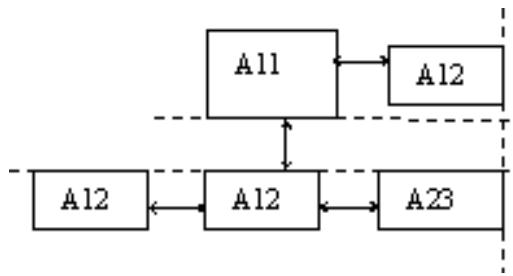
- Align two macro cells with pins – You can select this option when you want to create two-dimensional macro arrays and align numerous small macro cells that have uneven shapes.

For a rectilinear 2-D array with a horizontal row structure, the first part refers to how the rows are aligned, and the second part refers to how the cells in a row are aligned. For a rectilinear 2-D array with a vertical row structure, the first part refers to how the cells in a vertical row are aligned and the second part refers to how the rows are aligned.

The valid values for the option have two parts and are combined with a dash (-). They are “left-bottom,” “left-center,” “left-top,” “right-bottom,” “right-center,” “right-top,” “center-bottom,” “center-center,” and “center-top.” For a heterogeneous rectangular 2-D array, the first part—“left,” “right,” or “center”—refers to how cells in a column are aligned; the second part—“bottom,” “top,” or “center”—refers to how cells in a row are aligned.

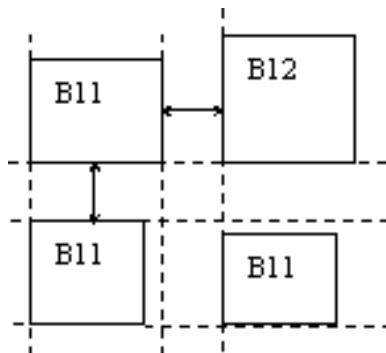
[Figure 5-8](#) shows the result of applying the constraint to align a two-dimensional rectilinear heterogeneous macro array.

Figure 5-8 Aligning a Rectilinear 2-D Heterogeneous Array



[Figure 5-9](#) shows the result of applying the constraint to align a two-dimensional rectangular heterogeneous macro array.

Figure 5-9 Aligning a Rectangular 2-D Heterogeneous Array

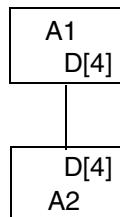


- Pin of macro 1 and Pin of macro 2 – You can align cells in macro arrays having two macro cells by aligning a reference pin (a pin on the first cell in the array, which is the reference cell) and a pin on the other cell in the array.

This option takes effect only if the array has two macro cells.

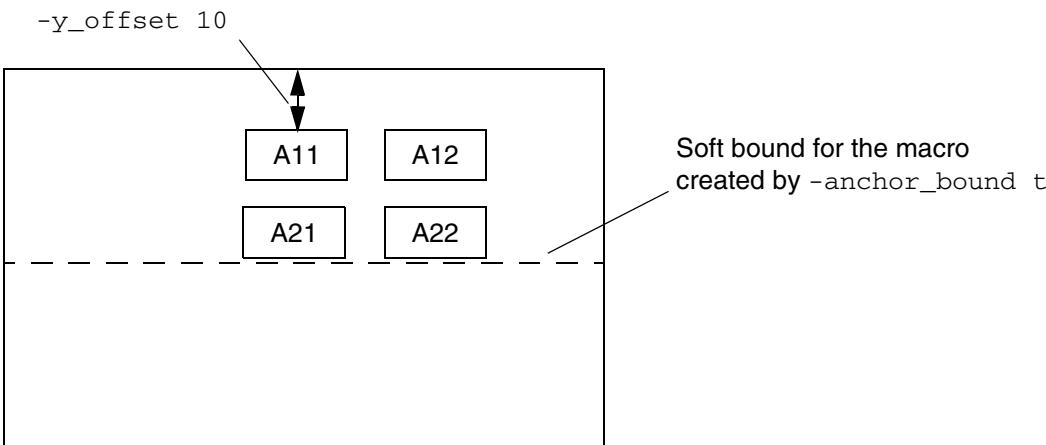
[Figure 5-10](#) shows the result of aligning cells in a macro array by using pins.

Figure 5-10 Result of Aligning Cells in Macro Arrays by Using Pins



[Figure 5-11](#) shows a two-dimensional array offset 10 microns from the top edge of the core area.

Figure 5-11 Two-Dimensional Array Offset 10 Microns From Core Top Edge



4. Click OK.

Setting Floorplan Placement Constraints On Macro Cells

During floorplanning, you can define placement constraints that restrict specified macro cells and macro arrays to particular regions, orientations, alignments, and so forth. This constrains the `create_fp_placement` command in placing the macro cells in the floorplan.

When you specify the constraints on macro cells and macro arrays, keep the following points in mind:

- The physical library and design must be loaded.
- Set these floorplan placement constraints before you run the `create_fp_placement` command.
- During placement, macro cells can be flipped or rotated according to legal orientation constraints from the library or by constraints you set.
- You must provide a list of macro cells to which the constraints apply. You can provide this list by using the `get_cell` command (for example, `[get_cell RAM]`). If you list a macro cell that is not found, an error occurs.
- To report the options that are set, use the `report_fp_macro_options` command. The report provides details about a particular macro cell or array or all the macro cells or arrays. Information includes the name of the macro cell or macro array, anchor bounds, offsets, and legal orientation for macro cells. (Legal orientation does not apply to macro arrays.)

To set floorplan placement constraints that restrict specified macro cells and macro arrays during virtual flat placement,

1. Choose Placement > Macro Constraints.

The Macro Constraints dialog box appears.

Alternatively, you can use the `set_fp_macro_options` command.

Each constraint is displayed with a unique name, type, and list of macros or macro arrays.

2. To create a new macro options constraint, select the Add button on the Macro Constraints dialog box.

From the pop-up menu choose Macro Options.

The Add Macro Options dialog box appears.

3. Set the options, depending on your requirements.

- Macro objects – Enter a list of macro cells to be constrained.
- Legal orientations – Select this option to restrict the orientation to a valid orientation that you specify. If you do not use this option, orientation is not restricted during placement. The default is enabled.

You can provide a single orientation to force the placement engine to place the macro cell in a fixed orientation. After the cell orientation is fixed, the placement tool cannot change the orientation.

The value can be one or more than one of the following orientations using the Design Exchange Format (DEF) syntax: N, W, S, E, FN, FW, FS, FE

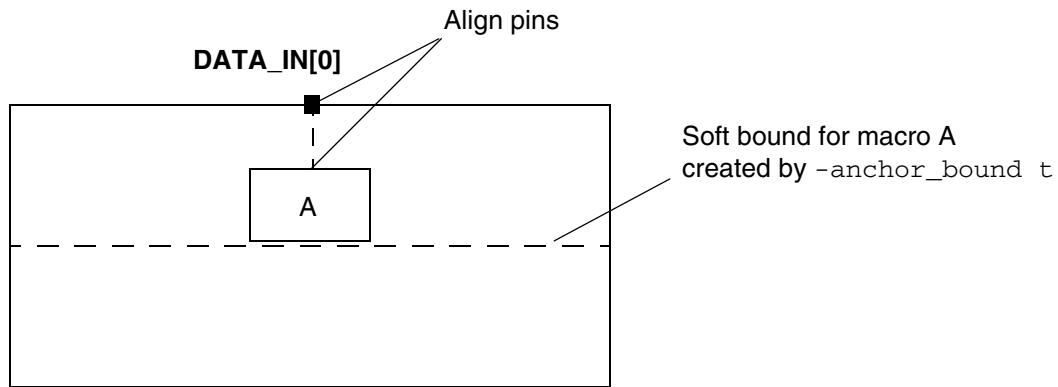
The orientation you specify overrides the legal orientations defined in the physical library. If you do not specify this option, the macro cell's legal orientation is used in the placement.

- Align Pins – You can align a specified port or pin to a pin of a constrained macro cell by using the this option. The argument is a list of two objects: reference port or pin followed by the constrained pin.

[Figure 5-12](#) shows the alignment between a port and a pin of a constrained macro cell.

To constrain macro A so that port DATA_IN[0] of the block aligns with DATA[0] of macro A, the DATA_IN pins were previously grouped so that they align with the other pins of macro A.

Figure 5-12 Aligning a Port to a Pin of a Constrained Macro Cell



- Anchor bound – You can set an anchor bound for macro placement to constrain the macro placement to the boundary region you specify, forcing the macro cell to stay at one of the boundaries of the block. To do this, select the “Anchor bound” option, which sets a soft bound. Because the bound is soft, placement can move the macro cell out of the bound if there is no space and the cost is too high.

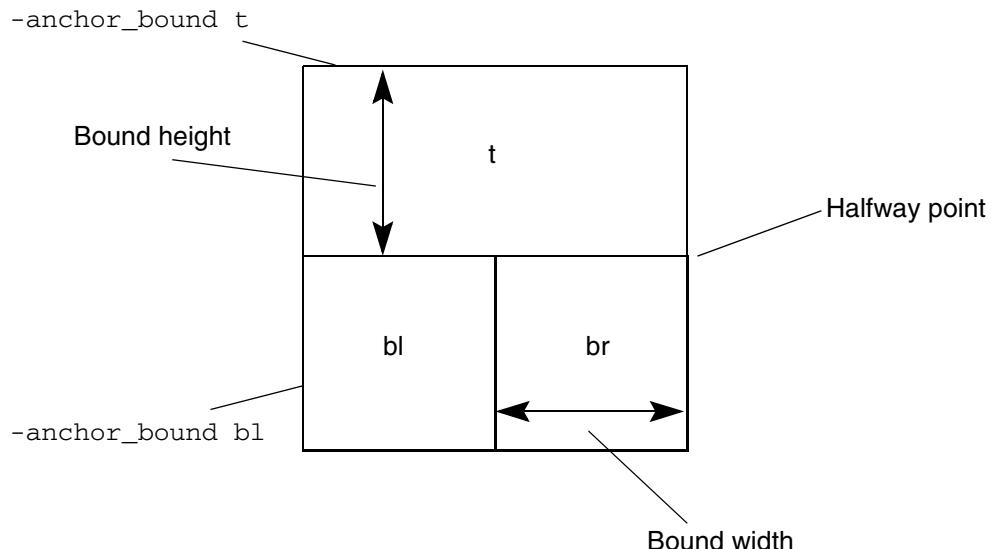
You can set one of the following 13 designations used for 16 boundary regions, the area in which the macro is placed.:

Keyword	Definition
br	bottom-right corner
tl	top-left corner
t	top
tr	top-right corner

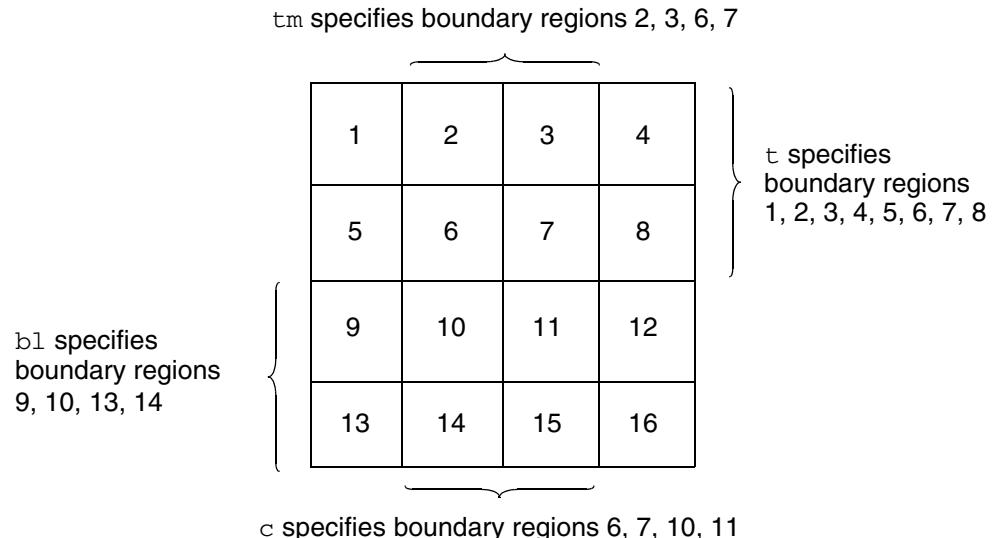
Keyword	Definition
r	right
b	bottom
bl	bottom-left corner
l	left
tm	top-middle
bm	bottom-middle
lm	left-middle
rm	right-middle
c	center

[Figure 5-13](#) shows boundary areas and values for the `-anchor_bound` option.

Figure 5-13 Boundary Areas and Values for the -anchor_bound Option



[Figure 5-14](#) shows the key for determining how to set the value for each of the 16 boundary regions.

Figure 5-14 Key for Determining Boundary Region Values for the -anchor_bound Option

The default bound area is

1/4 of the core area when you use bl, tl, br, tr, tm, bm, lm, rm, or c

1/2 of the core area when you use t, b, l, or r

The entire core area if you do not specify an anchor bound

As shown in [Figure 5-13 on page 5-12](#), specifying the anchor bound as t creates the move bound at the top-half of the core area, and specifying the anchor bound as bl creates a move bound at the quarter core area on the bottom-left area. If the half- or quarter-core area is not large enough to hold the macro cell in any dimension, the move bound is set to cell width or height in that dimension.

You can create two bounds with the same anchor point.

- X offset and Y offset – You can set a distance between the constrained macro and the core edges by selecting either the X offset or Y offset option or both options.

You must use the “Anchor bound” option when you use these options. The available offset options depend on the type of anchor bound, as listed in [Table 5-1](#).

Table 5-1 Anchor Bound and Offset Option

When the anchor bound is this	Use this offset
t, b, tm, or bm	Y offset
l, r, lm, or rm	X offset

Table 5-1 Anchor Bound and Offset Option

When the anchor bound is this	Use this offset
tl, tr, bl, br, or c	X offset or Y offset or X offset and Y offset

- Side channels – You can create a side channel by using this option. Specify the desired channel size on the left, right, top, and bottom sides of the chip, respectively, in microns. The macros are then placed at least the specified distances away from the core edges.

For example, to place a user defined macro array at least 20 microns away from the left edge, 30 microns away from the right edge, and 30 microns away from the top edge of the core area, enter 20 in the “Left” text box, 30 in the “Right” text box, 30 in the “Top” text box, and 0 in the “Bottom” text box.

4. Click OK.

Placing a Macro Cell Relative to an Anchor Object

You can set a relative location constraint on a cell, which is referred to as the target cell with respect to another cell, which is referred to as the anchor cell, or on a corner of the core area or plan group. This fixes the cell in a certain location with respect to the core area.

The anchor in a relative location constraint must be fixed. For example, this could be a corner of the core area or plan group, a fixed cell, or a cell that is anchored by some other relative location constraint.

Note:

If the relative location constraint is called more than once on a cell, the last one overrides any relative location constraint previously set on the cell.

To place a macro cell relative to an anchor object,

1. Choose Placement > Macro Constraints.

The Macro Constraints dialog box appears.

Alternatively, you can use the `set_fp_relative_location` command.

Each constraint is displayed with a unique name, type, and list of macros or macro arrays.

2. To create a new relative location macro constraint, select the Add button on the Macro Constraints dialog box.

From the pop-up menu choose Relative Location.

The Add Relative Location dialog box appears.

3. Set the options, depending on your requirements.

- Constraint name – Enter a name of the relative location constraint.
- Anchor object name – Enter the name of the anchor object. The object can be a plan group, a fixed macro cell, a macro cell that has its own relative location, or a core area.

Specify the corner of the anchor object's bounding box that the constraint uses. You can choose from the following designations: bl (bottom-left corner), br (bottom-right corner) tl (top-left corner), and tr (top-right corner).

- Target macro – Specify the name of the macro cell to which the constraint is applied.
- Orientation – Specify the orientation in which to place the target cell. You can choose from the following values: N, S, E, W, FN, FS, FE, and FW. The default is N.
- Corner – Specify the corner of the target cell on which to apply the constraint. You can choose from the following designations: bl (bottom-left corner), br (bottom-right corner), tl (top-left corner), and tr (top-right corner). The default is bl.
- X offset – Specify the distance in microns from the target cell to the anchor corner of the anchor cell in the X-dimension. You can enter a positive, negative, or zero value. The default is 0.
- Y offset – Specify the distance in microns from the target cell to the anchor corner of the anchor cell in the Y-dimension. You can enter a positive, negative, or zero value. The default is 0.

4. Click OK.

Reporting Relative Location Constraints

You can use the `report_fp_relative_locations` command to report macro relative location constraints set by the `set_fp_relative_location` command.

Extracting Relative Location Constraints

From an existing placement, you can extract location constraints for a list of macro cells (target cells) relative to an anchor object by using the `extract_fp_relative_location` command. The extracted constraints can be output to a file.

The `extract_fp_relative_location` command uses the following syntax:

```
extract_fp_relative_location -target cells cells
    [-target_corner bl | br | tl | tr] [-anchor_object object_name]
    [-anchor_corner bl | br | tl | tr] [-output_file file_name]
```

[Table 5-2](#) describes how to use the relative location constraints.

Table 5-2 Relative Location Constraints

To do this	Use this option
Specify the names of the macro cells whose locations will be extracted in the format of relative location constraints. The default is all macro cells. (Required)	<code>-target_cells cells</code>
Specify the corner of the target cells on which to extract the relative location constraints. You can choose from the following designations: bl (bottom-left corner), br (bottom-right corner), tl (top-left corner), and tr (top-right corner). The default is bl. (Optional)	<code>-target_corner</code>
Specify the name of the anchor object. The anchor object can be a plan group, a fixed macro cell, or a core area. The default is core area. (Optional)	<code>-anchor_object object_name</code>
Specify the corner of the anchor object's bounding box that will be used in extracting the constraints. You can choose from the following designations: bl (bottom-left corner), br (bottom-right corner) tl (top-left corner), and tr (top-right corner). The default is bl. (Optional)	<code>-anchor_corner</code>
Specify the name of the file into which the extracted constraints are output. The default is output to a console log view.	<code>-output_file file_name</code>

Removing Relative Location Constraints

You can remove relative location constraints set on the specified target cells and any other cells that are anchored to these cells by using the `remove_fp_relative_location` command. By default, all relative location constraints are removed.

The `remove_fp_relative_location` command uses the following syntax:

```
remove_fp_relative_location [-name constraint_name] [-target cells cells]
```

Using a Virtual Flat Placement Strategy

This section describes the constraints you can use to control the virtual flat placement. These constraints are not persistent in the Milkyway database.

They are grouped into four categories:

- Hard Macro

- Net Weighting
- Congestion
- Miscellaneous

Note:

You can set all the default values by using the `set_fp_placement_strategy -default` option.

Table 5-3 lists the hard macro control constraints.

Table 5-3 Hard Macro Control

Option	Description
<code>set_fp_placement_strategy -pin_routing_aware on off</code>	<p>Set this parameter to a value of on to provide pin aware information to virtual flat placement. Macro placement (<code>create_fp_placement</code>) takes this information into consideration and creates macro-only blockages along the edges of a plan group or a soft macro so that the macro cells do not block the edges of the plan groups or soft macros. This reserves enough space along the edges of the plan group or soft macro for pin assignment to place pins and for the router to route nets to the pins.</p> <p>The default is off.</p>

Table 5-3 Hard Macro Control

Option	Description
set_fp_placement_strategy [-macro_orientation automatic all N]	<p>Controls how macro orientations are determined.</p> <p>A value of <code>automatic</code> tells the IC Compiler tool to analyze the macro's reference library cell and determine a good orientation for each reference cell, based on the reference cell's pins. This is the default.</p> <p>A value of <code>all</code> rotates macros to any legal orientation during placement.</p> <p>A value of <code>N</code> means all macros will be in one of the following orientations unless overridden by a reference library constraint or a user-set macro orientation constraint.</p> <ul style="list-style-type: none"> 0 (0 degrees rotation) 180 (180 degrees rotation) 0-mirror (reflection in the y-axis) 180-mirror (180 degrees counter clockwise rotation, followed by a reflection in the y-axis)
set_fp_placement_strategy -macro_setup_only on off	<p>When set to <code>on</code>, the placement engine places all macro arrays outside the core area for viewing purposes. You can then manually place the macro arrays inside the core area if desired.</p> <p>Macros with relative locations constraints are still placed within the core area.</p> <p>Standard cells are not placed.</p> <p>The default is <code>off</code>.</p>

Table 5-3 Hard Macro Control

Option	Description
set_fp_placement_strategy -auto_grouping none user_only low high	<p>Controls how much automatic hard macro array packing will be done, based on design hierarchy information.</p> <ul style="list-style-type: none"> – Wire length is reduced if a single net drives the same pin of more than one macro cell in an array. – The number of scattered hard macros is reduced, increasing the routing area. – The <code>-auto_grouping</code> option is hierarchy-aware, which means that macros in the same physical hierarchy will be arrayed (grouped) together. – User-created arrays are honored.

Set the `-auto_grouping` option to `none` if you do not want to generate macro arrays. A value of `user_only` generates only user-defined macro arrays. A value of `low` (the default) generates macro arrays from small macros only, and a value of `high` generates macro arrays from all macros. (Note: You can also pack selected hard macros into a region. See “[Packing Hard Macros Into an Area](#)” on page 5-37.)

Table 5-3 Hard Macro Control

Option	Description
set_fp_placement_strategy -macros_on_edge on off	Creates a placement with hard macros on the boundary edge of the chip or plan group edge. This option eliminates the fragment spaces that are created when hard macros are scattered, thereby creating a large open space in the core area for placement and routing.
	Set this option to <code>on</code> if a large percentage of the chip area contains hard macros. Moving hard macros to the edge will reduce congestion and improve routability.
	For a hierarchical design, set this option to <code>off</code> until you have shaped the plan groups and placed them inside the core area of the chip.
	This option interacts with the hierarchy gravity option. If the hierarchy gravity option is off, the macros are placed on the edges of the chip. If the hierarchy gravity option is on, the macros are placed on the edges of the physical hierarchical boundaries to which they belong.

Table 5-3 Hard Macro Control

Option	Description
<code>set_fp_placement_strategy -sliver_size distance</code>	Defines the minimum channel size that can be populated by standard cells. A sliver is a channel that is too small for placement of any standard cells. For example, if a <code>sliver_size</code> is 10, a channel of height 9 would not have any standard cells inside it. Slivers apply to channels created between any two of the following objects, provided that the object in question occupies at least 1/100 of the total chip area: fixed or movable hard macros, placement blockages, plan group edges, or core area edges. (Note that the keepout margin of hard macros is not included in the sliver size. Hard macro slivers are measured from the edge of the keepout margin, not from the edge of the macro.)
	During placement, blockages are dynamically created over narrow channels between macros as well as over channels between macros and chip edges.
	Use this option if there is congestion in narrow channels and if channels contain standard cells. If there are several channels containing standard cells that have a lot of congestion, set the <code>sliver_size</code> to the largest width of those channels.
	Legal values are any real numbers greater than or equal to 0.0 (the default).

Table 5-3 Hard Macro Control

Option	Description
<code>set_fp_placement_strategy -fix_macros none soft_macros_only all</code>	Controls whether macros should be temporarily fixed in position during virtual flat placement. When set to <code>none</code> (the default), all macros are allowed to move during placement.
	When set to <code>soft_macros_only</code> , soft macros are fixed during virtual flat placement. The <code>create_fp_placement</code> command will not move them, regardless of their placement status in the Milkyway database.
	When set to <code>all</code> , all soft macros and hard macros are fixed during virtual flat placement.
	Use this option if your hard macros are in good positions in the design and you want to fix them during virtual flat placement.
	This parameter does not change the “fixed” attributes on macros.
<code>set_fp_placement_strategy -snap_macros_to_user_grid on off</code>	A value of <code>on</code> snaps the lower-left corner of the hard macro’s bounding box to the grid points defined by the <code>set_user_grid</code> command. After you set this value, during any future virtual flat placement, each hard macro is placed (snapped) with its lower-left corner on a grid point.
	The default is <code>off</code> .

Table 5-4 lists the net weighting control constraints.

Table 5-4 Net Weighting Control

Option	Description
<pre>set_fp_placement_strategy -voltage_area_interface_net_weight float</pre>	<p>You can use this control option to set a weight that is higher than normal on the interface nets of voltage areas, resulting in more effort to shorten these nets. The placement engine uses this net weight to “pull” the cell toward the interface boundary of the voltage area.</p> <p>The value is a floating-point number from 0.0 to 10.0.</p>
<pre>set_fp_placement_strategy -voltage_area_net_weight_LS_only on off</pre>	<p>When set to <code>on</code>, a net weight is set only on interface nets connected to level-shifter cells and only during the placement of the level shifters. This prevents non-level-shifter nets from being unintentionally pulled to the edge of the voltage area.</p> <p>The default is <code>off</code>.</p>
<pre>set_fp_placement_strategy -IO_net_weight</pre>	<p>Controls the special weighting on I/O nets. This option is useful if the design requires a short distance between the I/Os and their connected standard cells or hard macro cells. The allowed range is from 0.0 through 10.0. The default is 1.0.</p> <p>Example: <code>IO_net_weight = 2.0</code></p> <p>The nets connected to I/Os are weighted twice as high as the other nets, and therefore, are likely to be short.</p> <p>Example: <code>IO_net_weight = 0.0</code></p> <p>The lengths of the nets are ignored. This is useful if no final I/O placement has been done and you want the placer to help determine the I/O locations.</p>

Table 5-4 Net Weighting Control

Option	Description
<code>set_fp_placement_strategy -plan_group_interface_net_weight</code>	Controls the net weight on interface nets that cross the edges of exclusive plan groups. This option is useful if the cells in your design need to be pulled closer to the plan group boundary when those nets cross the boundary. The allowed range is from 0.0 through 10.0. The default is 1.0.

[Table 5-5](#) lists the congestion control placement constraints.

Table 5-5 Congestion Control

Option	Description
<code>set_fp_placement_strategy -congestion_effort low high</code>	Controls the effort level of congestion-driven placement. When set to <code>high</code> , the amount of padding is increased during the placement, thereby reducing the congestion. The default is <code>low</code> .

[Table 5-6](#) lists miscellaneous placement constraints.

Table 5-6 Miscellaneous Control

Option	Description
<code>set_fp_placement_strategy -legalizer_effort low high</code>	Controls the effort level of the legalizer. Set this option to <code>low</code> if the legalizer is running too long. The default is <code>high</code> .
<code>set_fp_placement_strategy -spread_spare_cells on off</code>	Spreads spare cells that have no connection to any other cell evenly across the chip. If you set this option to <code>off</code> , the placement engine does not do anything special to spread the spare cells. The default is <code>on</code> .

Table 5-6 Miscellaneous Control

Option	Description
set_fp_placement_strategy -virtual_IPO on off	<p>Controls whether to use virtual in-place optimization during timing-driven placement. Timing-driven virtual flat placement uses a net weighting method. It adds a higher weight on the more critical nets, based on critical path information from the timing analysis. Some nets, however, that might be critical at the time of placement may not be considered critical later on, because in-place optimization can easily fix the timing on these nets and paths. Using virtual in-place optimization during timing-driven virtual flat placement estimates the effects of timing optimization so that higher net weights are applied to the real critical nets.</p>

Use the `report_fp_placement_strategy` command to display floorplan-related option values.

Creating Macro Blockages for Hard Macros

If you do not want hard macros to be placed in certain areas of your design, you can create a macro blockage to restrict the placement of hard macros.

For example, there might be a channel between two plan groups or two hard macros that is large enough for standard cells to go through, but if a hard macro is placed in the channel, routing would be adversely affected. In this case, placing a macro blockage in the channel restricts the placement of the hard macro. Another example might be if a hard macro were

to be placed near a power pad, which could have an adverse effect on the IR drop. In this case, placing a macro blockage around the power pad ensures that the hard macro is not placed close to the power pad.

To create a placement blockage for hard macros only,

1. Choose Floorplan > Create Placement Blockage.

The Create Object Tool (Placement Blockage) dialog box appears.

Alternatively, you can use the `create_placement_blockage` command.

2. Enable the Hard Macro option to create a macro blockage.
3. Click in the Name text box, and enter a unique name for the macro blockage.
4. Specify the bounding box (rectangle) in which to create the blockage. The rectangular coordinates are relative to the current design and identify the lower-left and upper-right corners of the rectangular area.

You can also specify the area graphically in the layout window by selecting the button next to the Coordinates field.

5. Click OK or Apply.

Padding the Hard Macros

To avoid placing standard cells too close to macros, which can cause congestion or DRC violations, you can set a user-defined padding distance or keepout margin around the macros. You can set this padding distance on a selected macro's master cell. During virtual flat placement, no other cells will be placed within the specified distance from the macro's edges.

To set a padding distance (keepout margin) on a selected macro's master cell,

1. Choose Placement > Set Keepout Margin.

The Set Keepout Margin dialog box appears.

Alternatively, you can use the `set_keepout_margin` command.

2. Specify the type of keepout margin to be created, either hard or soft. The default is hard
3. Choose whether to set a user-defined padding distance on all macros (the default), on specified macros, or on specified instances. Enter the names of the specified masters or instances, if applicable.

4. If you selected “All macros,” you can choose to either derive the amount of padding based on the number of pins and track width, or you can specify explicit keepout distances on all four sides. If you select “Specified masters” or “Specified instances”, you must specify explicit keepout distances on all four sides.

If you choose “Derive outer keepout coordinates”, specify the following:

- The number of tracks per macro pin.

If you have macros with lots of pins, it is recommended that you set the padding to 0.5 to 1.0 track per pin (0.5 reserves one track for every two pins) to reserve some routing area around the macros and give the router enough space to route to the macro’s pin.

- The minimum padding per macro size.

This is useful when there are a few pins on some sides of some macros. The default is 0.

- The maximum padding per macro size.

The default is -1, which means that the placer would apply a 40x metal1 pitch limit on all macros.

If you choose “Use specified outer keepout coordinates”, enter the explicit padding in microns in the Top, Bottom, Right, and Left fields.

5. Click OK or Apply.

After the padding distance or keepout margin is defined on a master cell, it will follow the movement of the hard macros; that is, when a hard macro moves, the padding moves along with it.

The padding is automatically saved to the design database when the design is saved.

Automatic Padding

If you do not define a padding distance (keepout margin), the tool automatically generates a default keepout margin based on the pin count on the edges for all the hard macros. By default, the placement keepout margin on an edge is calculated as follows:

$$(\text{number of pins on edge } /2)^* \text{ pitch}$$

Note:

You can change this value by changing the “Tracks per macro pin” option value in the Set Keepout Margin dialog box.

User-defined padding overrides automatic padding.

Placing Hard Macros and Standard Cells

You can perform a virtual flat placement and obtain a flat placement of the hard macros and standard cells. Based on the placement results, you can then decide the relative locations, shapes, and sizes of the top-level logic blocks. If you have a design with plan groups or voltage areas, the refined placement honors the exclusiveness of the plan groups and voltage areas.

Note:

The main consideration when placing a large hard macro is congestion; wire length becomes secondary. A hard macro is considered large if it takes up 5 percent or more of the chip area or 40 percent or more of the chip width. To improve routability, place large hard macros at the edges of the chip boundary.

To place the hard macros and standard cells simultaneously,

1. Choose Placement > Place Macros and Standard Cells.

The Place Macros and Standard Cells dialog box appears.

Alternatively, you can use the `create_fp_placement` command.

2. Click Default, and then click OK.

This section includes the following topics:

- [Planning the Location and Shape of Voltage Areas](#)
- [Controlling the Placement](#)
- [Moving Cells In the Core Area to a New Location](#)
- [Packing Hard Macros Into an Area](#)
- [Manually Adjusting the Hard Macros](#)
- [Performing Simultaneous Placement and Pin Assignment for Block Level Designs](#)

Planning the Location and Shape of Voltage Areas

You can automatically place and shape voltage areas, including voltage areas that are physically nested. You can also create voltage areas outside the chip area.

This section includes the following topics:

- [Handling Nested Voltage Areas](#)
- [Supported Voltage Area Physical Boundary Scenarios](#)

- [Updating Voltage Areas in a Design](#)
- [Removing Voltage Areas](#)

In IC Compiler, you can define voltage areas at any level of a logic hierarchy. Both rectangular and rectilinear voltage areas are allowed. Voltage areas cannot consist of disjoint rectilinear and rectangular shapes.

Power domains and voltage areas should maintain a one-to-one mapping relationship. A power domain and its matching voltage area should have the same name and same set of hierarchical cells. When a power domain is mapped with the voltage area, the associated logic information is automatically derived from the power domain.

Voltage areas can be explicitly associated with existing power domains, or they can be created without specifying any explicit association with power domains. They can also be created when no power domains have been defined for the design.

A given voltage area can contain one or more physically nested voltage areas (See [“Handling Nested Voltage Areas” on page 5-31](#).) A voltage area must completely contain its nested voltage areas. These nested voltage areas correspond to logically nested power domains. Voltage areas that would overlap physically must be resolved into nonoverlapping, abutted rectangular or rectilinear subareas. Intersecting voltage areas are not supported.

Planning the location and shape of voltage areas requires the following steps:

1. Create a voltage area of a specific geometry on the core area of the chip.

Choose Floorplan > Create Voltage Area

The Create Voltage Area dialog box appears.

Alternatively, you can use the `create_voltage_area` command.

- Name – Specify the name of the voltage area you want to create. The `DEFAULT_VA` is reserved and cannot be used.

If you specify this option, you must also provide a list of hierarchical cells that are contained within the voltage area. The logic hierarchies are assigned to specific voltage areas when you create or update these voltage areas

- Power Domain – If your design contains power domains, you can use this option with the corresponding power domain name to create a voltage area. Then the power domain name is also the voltage area name. After the voltage area is created, the power domain contains the corresponding boundary information.

In this case, do not specify a list of hierarchical cells. The cell list is provided in the power domain definition. Power domains and voltage areas should maintain a one-to-one mapping relationship. A power domain and its matching voltage area

should have the same name and same set of hierarchical cells. IC Compiler places these cells in the given voltage area. Using the power domain option ensures the correct alignment of the logic hierarchies with the voltage areas.

- Horizontal guard band width and Vertical guard band width – You can use guard bands to ensure that no shorts occur at the boundaries of the voltage areas. Guard bands define hard keepout margins surrounding the voltage areas. No cells, including level shifters and isolation cells, can be placed within the guard band margins.

For example, using guard bands is recommended when the rows are the same for all the voltage areas. The guard bands guarantee that the cells in different voltage areas are separated so that power planning does not introduce shorts.

You can specify a guard band width in the horizontal and vertical direction when creating or updating voltage areas.

- Target utilization – Specifies the target utilization for the voltage area during shaping.

You must specify a utilization value between 0.1 and 1.0. The default utilization is the same value used for the rest of the design. Larger utilization values result in smaller voltage areas, with less extra space allowed for routing. A voltage area is created outside the chip area using the given utilization with value of 1.0.

To change the target utilization, you must use the `remove_voltage_area` command to remove all voltage areas or specified voltage areas from the design, and then re-create the voltage area and the new target utilization value.

- Coordinates – If you know an ideal location for a voltage area, you can direct it by specifying explicit coordinates. The voltage area geometry can be a rectangle or a rectilinear polygon.

To define a rectangular voltage area, select the rectangular-shaped button, and type the x- and y-coordinates for the upper left and lower right corners of the rectangle in the Coordinates box.

To define a rectilinear voltage area, select the rectilinear-shaped button, and type the x- and y-coordinates for each corner of the polygon in the Coordinates box.

After the user-defined voltage areas are created, the tool automatically derives a default voltage area for placement of cells not specifically assigned to any voltage areas.

- Set as fixed – You can place newly created voltage areas inside the core in fixed locations. The voltage areas with a fixed attribute are ignored by the `shape_fp_blocks` command and are not reshaped. By default this option is off.
- Color cell instances – (Optional) Select a method for coloring cells in the voltage area.

To disable cell coloring, select None.

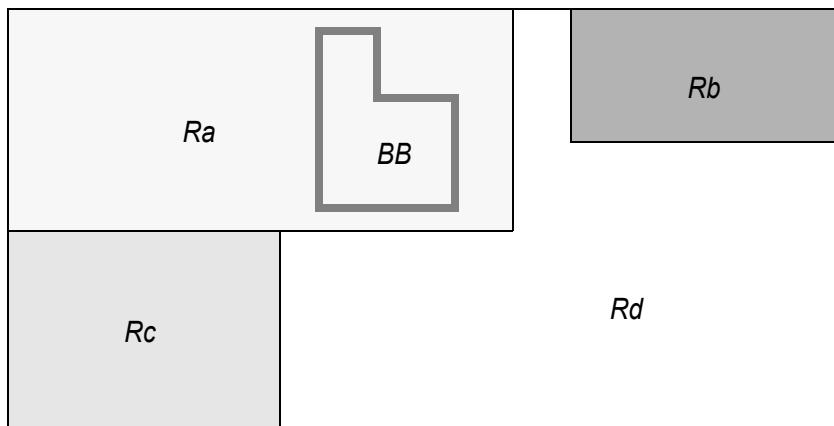
To cycle through the available colors, select Cycle. This is the default.

To apply a specific color, select Specified and select a color in the color list.

2. Click OK or Apply.
3. Use the `create_fp_placement` command to place the design, keeping voltage area cells together. You can also choose Placement > Place Macro and Standard Cells. (See “[Placing Hard Macros and Standard Cells](#)” on page 5-28.)
4. Use the `shape_fp_blocks` command and options to automatically shape and place the voltage area boundaries. You can also choose Placement > Place and Shape Plan Groups. (See “[Automatically Placing and Shaping Objects In a Design Core](#)” in [Chapter 6](#).)

[Figure 5-15 on page 5-31](#) shows a multivoltage design divided into several voltage areas.

Figure 5-15 Multivoltage Design With Several Voltage Areas



In this design, the floorplan has four voltage areas: Ra, Rb, Rc, and Rd. The default voltage area is Rd. Each voltage area has a corresponding logical partition, A, B, C, and D, that aligns with its respective voltage area. A hard bound can be defined inside a voltage area but not across voltage areas. In this example, a hard bound, BB, is defined inside voltage area Ra.

Special always-on site rows can be defined within a given voltage area. The standard cells placed in these site rows serve as the always-on cells of the shut-down power domain associated with the given voltage area.

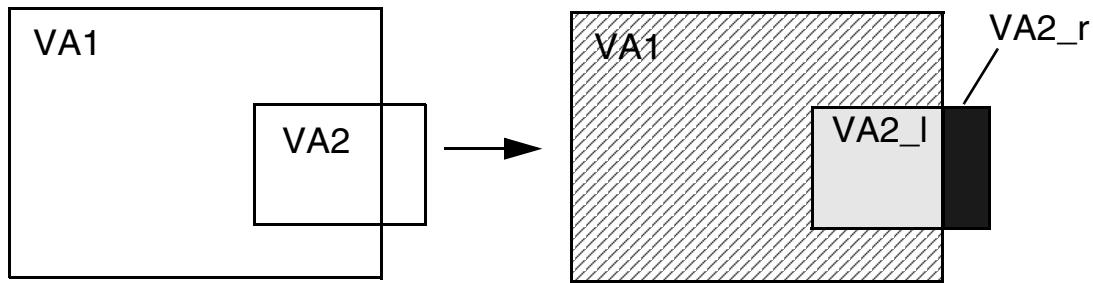
Handling Nested Voltage Areas

Voltage areas can be physically nested corresponding to the nested relationship of the logic hierarchy of nested power domains. You still use the `create_voltage_area` command to define the individual voltage areas. A given voltage area can contain one or more nested voltage areas. Note that a voltage area must completely contain its nested voltage areas. Intersecting voltage areas are not supported.

In the case where logically nested hierarchies would lead to overlapping voltage areas, you can use the `create_voltage_area` command to resolve the overlapping areas by defining a number of voltage areas consisting of separate, *abutted* rectangles or rectilinear shapes. Each shape would share at least one boundary with another shape of the given voltage area.

[Figure 5-16](#) shows an example of areas intersecting and how you might resolve them into three physically abutted, separate voltage areas, VA1, VA2_l, and VA2_r. Voltage area VA1 is an eight-sided rectilinear area, and voltage areas VA2_l and VA2_r are four-sided rectangles.

Figure 5-16 Example of Physically Resolved Voltage Areas

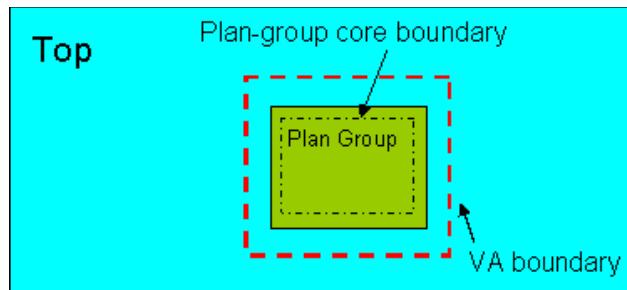


Supported Voltage Area Physical Boundary Scenarios

The following voltage area (VA) physical boundary scenarios are supported.

- The voltage area boundary encloses the plan group core boundary, as shown in [Figure 5-17](#).

Figure 5-17 Voltage Area Boundary Encloses Plan Group Boundary

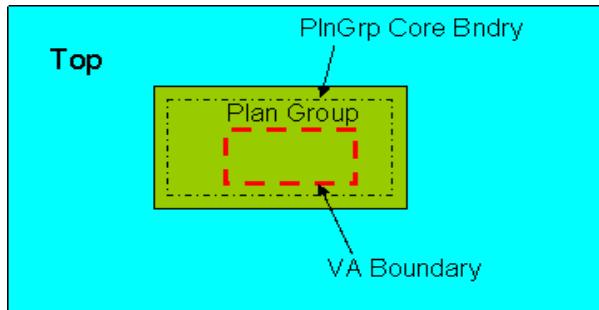


During commit hierarchy (`commit_fp_plan_groups` command), the physical voltage area boundary is not copied down into the soft macro, and child cell instances are not assigned logically to any non-`DEFAULT_VA`. The power domain is updated to swap the hierarchical cell with the new cell instance.

During uncommit hierarchy (`uncommit_fp_soft_macros` command), the soft macro child cell instances are pushed back up to the top CEL view, and are reassigned logically to the original top cell non-DEFAULT_VA.

- The voltage area boundary is contained by the plan group core boundary, as shown in [Figure 5-18](#).

Figure 5-18 Voltage Area Boundary Contained by Plan Group Boundary



During commit hierarchy, the physical voltage area boundary is copied down into the soft macro, and the child cell instances are assigned logically to their respective non-DEFAULT_VA's. The effects on associated power domains are not considered.

During uncommit hierarchy, the soft macro child cells are assigned logically to the original top cell non-DEFAULT_VA's.

Updating Voltage Areas in a Design

After you have created voltage areas and associated hierarchical cells with them, you can use the `update_voltage_area` command to add or remove cells from specified voltage areas. You can use the `get_voltage_area` command instead of explicitly specifying voltage area names. By adding or removing cells, you can adjust the utilization of the voltage area. However, you might prefer to change the size of the voltage areas and keep the utilization the same.

Removing Voltage Areas

Use the `remove_voltage_area` command to remove all voltage areas or specified voltage areas from the design. This includes removing any move bounds and guard bands belonging to the voltage area. If you remove a logically nested voltage area, the subhierarchy logic inherits the voltage properties of the parent hierarchy.

Controlling the Placement

This section describes the placement constraints that you can use to place hard macros and standard cells and control the placement results.

Choose Placement > Place Macro and Standard Cells. The Place Macro and Standard Cells dialog box appears.

Alternatively, you can use the `create_fp_placement` command.

- Selecting an effort level

The effort level controls the trade-off between the QoR and the amount of CPU runtime spent performing an initial flat placement. Select the low-effort option (the default) for very fast results that provide good plan group locations and hard macro locations.

Raising the effort level to high improves the quality of the placement, but incurs an increase in CPU runtime.

- Using the “Congestion driven” option

You can perform congestion-driven placement by selecting the “Congestion driven” option. The default is off. Standard cells and hard macros are placed together to minimize wire length and congestion in the design and to avoid cell overlaps.

Congestion-driven placement:

- Reduces congestion by adding padding to cells in congested areas.

The amount of padding needed for each cell depends on the congestion around each cell.

- Moves hard macros and standard cells in congested areas.
- Pads macros in automatically generated arrays, thereby changing the size of the array.

- Using the “Timing driven” option

You can perform incremental timing-driven placement to improve the timing result for your design by selecting the “Timing driven” option. The default is off.

Timing-driven placement:

- Improves the timing by moving cells on critical paths closer together.
- Moves hard macros on critical paths.

- Using the “Hierarchical gravity” option

The “Hierarchical gravity” option tends to keep cells of a hierarchical design logic block physically together in the chip layout. This type of grouping usually results in better placement because designs tend to partition well along hierarchical boundaries.

However, you should deselect this option if you know that the logic hierarchy does not represent a good physical partitioning of the design. This allows the placement engine the flexibility to place cells of a hierarchical block far from other cells from the same block. The default is on.

If there are no user-created plan groups, the placement engine analyzes the logic hierarchy of the design and uses the blocks with reasonable sizes as the default plan groups.

The use of this option is also recommended for placement consistency. For example, if you make a minor change to your design but keep the same cell instances together and then use this option, those cell instances will remain near their original locations.

- Choosing a maximum fanout

Enter a maximum fanout as a positive integer. The wire lengths of any nets with a fanout higher than the number you specify are ignored by the placement. The default is 512.

- Performing an incremental placement

If you want the placement to start with existing cell locations and incrementally improve on them during the placement, select the “Incremental” option. Use this option if you want to start with a good relative placement but do not want the cells to move too far from their current locations. (If cells need to be moved far from their current locations and you use this option, the result will be a poor placement.) This option is off by default, and the placer ignores existing cell locations and creates a placement from scratch.

Another reason to use this option would be if you have already run a placement on the design with placed plan groups inside the core area and there is a congestion or timing problem.

When plan groups are placed inside the core area, enable the Incremental option and then select from the following three suboptions to further refine the placement on selected parts of your design.

All – Runs incremental placement on all the standard cells and hard macros in the design. This is the default.

Top level cells only – Runs incremental placement only on the top-level cells that do not belong to a plan group.

Specified plan groups – Performs a refined placement on only the selected exclusive plan group cells. You can select one or more plan groups.

Specified voltage areas – Performs a refined placement on standard cells and hard macros within the specified voltage areas. The virtual flat placement engine will recognize voltage areas as physical constraints in the floorplan.

To create a floorplan with multiple voltage areas, you need to specify each logic module that belongs to the separate top-level power domains. Placement voltage areas are used to constrain the placement of cells to certain locations, based on the cell’s power domain. A power domain is a set of cells with the same voltage requirements. This means that each delineated power domain should include only cells driven from the same power supply.

In a single-voltage design, the entire design operates under the same voltage, the default voltage. In a multivoltage design, you need to define areas where all cell instances use a higher or lower voltage than the default voltage. The rest of the design area (outside of any defined voltage areas) uses the default voltage. For multivoltage designs, the default voltage area equals the design boundary minus all the voltage areas.

- Legalizing the resulting placement

Select the “Legalize resulting placement” option to resolve cell placement conflicts after doing initial placement. Overlaps are removed, and standard cells are placed at legal sites. Legalization is appropriate for the final placement. The default is on.

Alternatively, you can use the `legalize_fp_placement` command.

- Optimize pins

Select the “Optimize pins” option to perform simultaneous placement and pin assignment for block-level designs. The placement engine places the cells near the port locations according to the TDF pin constraints (side, side and order, and side and location) that you have set. The result is a simultaneous virtual flat placement and pin assignment for block level designs with a smaller wire length of nets connected to the constrained pins

- Ignore scan chain connectivity

Select this option if you want the placement engine to ignore scan chain connections during virtual flat placement.

- Specifying the number of CPUs

Enter the number of CPUs to be used in parallel during initial virtual flat placement. The number you enter should be an integer value less than or equal to the number of free CPUs on your machine. The default is one CPU.

Moving Cells In the Core Area to a New Location

You can move (unplace) macro cells, standard cells, or all cells that reside in the core area of the chip, and place them in a new location by using the `remove_placement` command. The command does not unplace cells with `fixed` or `dont_touch` attributes assigned to them.

The `remove_placement` command uses the following syntax:

```
remove_placement
  -object_type standard_cell | macro_cell | all
  -new_location top_right | origin | center
```

For the `-object_type` option, the default moves (unplaces) all cell types.

For the `-new_location` option, the default moves the cells to the `top_right`. Macro cells are moved to top side of the chip, and standard cells are moved to the right side of the chip.

If you specify the `-origin` argument, the standard cell coordinates are set to (0,0).

If you specify the `-center` argument, the standard cell coordinates are set to the center of the chip.

Packing Hard Macros Into an Area

You can specify an area (region) after performing virtual flat placement and automatically pack selected hard macros along the edges of that selected area. A data structure is automatically built for packing the hard macros.

To automatically pack hard macros into an area,

1. Choose Placement > Pack Macro Cells.

The Pack Macro Cells dialog box appears.

Alternatively, you can use the `pack_fp_macro_in_area` command.

2. Set the options, depending on your requirements.

- Pack all macro cells

Select this option (the default) to pack all macro cells inside the core area. Any top-level macros are also packed into the remaining area.

- Macro cells of specified plan groups

All hard macros belonging to selected plan groups are packed inside the plan group boundaries. If a plan group boundary is outside the core area, any macros belonging to that plan group are ignored. Top-level macros and macros belonging to other plan groups are also ignored.

- Specified macro cells

Select this option to pack specified macro cells into the core area.

If a polygon or rectangle is selected (it can be on any layer except a blockage layer) along with some hard macros, the hard macros are packed into the selected polygon or rectangle.

- Preferred edges

You can specify the edge of the area (region): L (left), R (right), B (bottom), and T (top) against which to pack the hard macros. These are the preferred edges for placing the macro cells. For example, if you select Left, Top, and Bottom, the macro cells will be placed along those edges first, arranging them in a “C” shape.

Selecting all four edges will cause the macro cells to be placed along all four edges, arranging them in an “O” shape. This is different from specifying no edges, which will cause placement to be based on other considerations such as wire length.

3. Click OK or Apply.

Manually Adjusting the Hard Macros

To manually adjust the hard macros, it is recommended that you observe the following principles, in addition to applying your own criteria:

- Follow the given relative locations from the hard macro placement command (`create_fp_placement`).
- Avoid isolated standard cell placement areas enclosed by a ring of hard macros.
- Avoid narrow standard cell placement areas in channels (slivers) between hard macros.
- Align similar hard macros.
- Push large hard macros to the core boundary.
- For standard cell placement, create a rectangular placeable area.

The “Align objects” and “Distribute objects” buttons on the Edit toolbar are particularly helpful for adjusting your hard macros.

Using the Align Objects Button

To use the “Align objects” button, first select one or more hard macros in the layout window, then move your cursor over the arrow next to the button, and click to display the left, right, top, bottom, vertical, and horizontal alignment icons. (

Alternatively, you can choose **Edit > Align** from the GUI menu, or you can use the `align_objects` command.

Selecting the right-align icon, for example, moves all selected hard macros so that their right edges are aligned. If one hard macro is selected, its right edge is aligned with the right edge of the core. If more than one hard macro is selected, one of those hard macros is also selected as the anchor point and the right edge of the remaining hard macros is aligned with the anchor object’s right edge. If any selected objects are fixed, the rightmost fixed hard macro is the anchor object; otherwise, the rightmost hard macro’s edge is the anchor edge.

Using the Distribute Objects Button

To use the “Distribute objects” button, first select one or more hard macros in the layout window, then move your cursor over the arrow next to the button, and click to display the left, right, top, and bottom distribution icons.

Alternatively, you can choose Edit > Distribute from the GUI menu, or you can use the `distribute_objects` command.

Selecting the left distribute icon, for example, moves all selected hard macros so that they are laid out next to each other from left to right. The distance between the hard macros is specified by the value in the Offset box.

Performing Simultaneous Placement and Pin Assignment for Block Level Designs

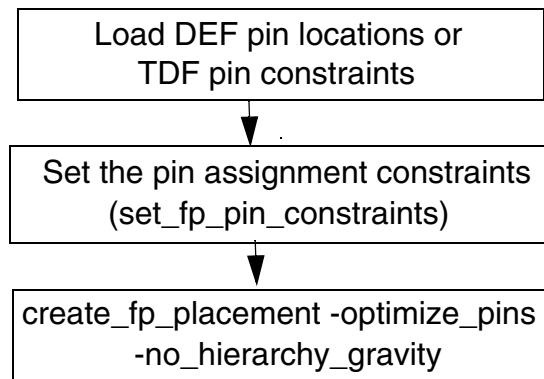
You can simplify your design flow by performing simultaneous placement and pin assignment for block level designs. To do this, choose Placement > Place Macro and Standard Cells > Advanced Options. Select “Optimize pins” on the dialog box.

Alternatively, you can use the `create_fp_placement -optimize_pins` command.

The placement engine places the cells near the port locations according to the TDF pin constraints (side, side and order, and side and location) that you have set. The result is a virtual flat placement and pin assignment for block level designs. Therefore, the overall quality of simultaneous placement and pin assignment is superior to running placement and pin assignment separately. The quality is measured by a smaller wire length of nets connected to the constrained pins.

[Figure 5-19](#) shows the design flow for the simultaneous placement and pin assignment for block level designs.

Figure 5-19 Simultaneous Placement and Pin Assignment Flow for Block Level Designs



Supporting Relative Placement Groups in Initial Virtual Flat Placement

You can use the `create_fp_placement` command to place cell instances in each relative placement group during initial virtual flat placement. By using the `create_fp_placement` and `create_placement` commands to place the cells in each relative placement group in the same way, the `create_fp_placement` command can provide you with better wire length and congestion estimates for your floorplan.

To achieve a good correlation between the `create_fp_placement` and `create_placement` commands when placing the relative placement cells, the following conditions apply, based on the current behavior of the `create_placement` command.

- If the cell instances in one relative placement group belong to different move bounds, the `create_fp_placement` command ignores this relative placement group. This also applies to plan groups automatically extracted by the `create_fp_placement` command, based on the logical hierarchy.
- No cell instances are placed over relative placement keepouts.
- If any cell instance in one relative placement group is fixed, this relative placement group is ignored by the `create_fp_placement` command and the cells are treated as nonrelative placement cells.

You use the `create_rp_group` command to create relative placement groups. A relative placement group is an association of cell instances, other hierarchical relative placement groups, and placement keepouts. A group is defined by the number of rows and columns that it uses. Use the `add_to_rp_group` command to add leaf cells to a relative placement group. Information about the relative placement groups is stored in the Milkyway database by using the `write_rp_groups` command. It contains the height and width of the top-level relative placement groups and the x and y offsets of the cell instances, placement keepouts, and hierarchical relative placement groups for each top-level relative placement group. The x and y offsets are used to anchor the relative placement cell instance with respect to the lower-left corner (the relative placement's origin) of its corresponding relative placement group.

The `create_fp_placement -no_legalize` command supports two types of relative placement groups, including the macro-only and standard-cell-only relative placement groups. The virtual flat placement stage allows the following relative placement objects: leaf cells, keepouts (hard, soft, and space), hierarchical groups, and relative placement cells to occupy multiple column and row positions.

During the design planning flow, you can perform the following relative placement functions:

- Apply compression by using the `-compress` option.

- Specify the anchor location by using the `-x_offset` and `-y_offset` options.
- Specify the utilization percentage by using the `-utilization` option.
- Ignore the relative placement group by using the `-ignore` option.

However, you should not use the following relative placement functions:

- Specify the group alignment pin by using the `-pin_align_name` option.
- Specify the right alignment by using the `-alignment bottom_right` option.
- Allow keepouts over tap cells by using the `-allow_keepout_over_tapcell` option.
- Legalize individual objects in a relative placement group.

Using the `create_fp_placement` Command to Place Cells in Relative Placement Groups

By default, relative placement is supported during initial virtual flat placement. The `create_fp_placement` command extracts information about the relative placement groups from the Milkyway database. A dummy cell, which contains both macros and standard cells in one relative placement group, is created for each top-level relative placement group. The Conjugate Gradient Placer (CGPL) places all cell instances in the flattened top-level relative placement groups together during initial virtual flat placement.

The `create_fp_placement` command honors the specified placement keepouts when it places the cell instances in the relative placement group, and no cell instances in the relative placement group are placed over the keepout areas.

Note:

The relative placement group's Milkyway properties are modified, based on the new locations of the placed dummy cells.

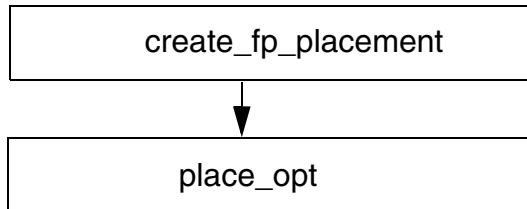
All relative placement cell instances are temporarily fixed. Any dummy cells related to the top-level relative placement groups are removed. This allows the `create_fp_placement` command to place other standard cell instances over the free spaces inside the relative placement bounding boxes.

Default Relative Placement Flow

Use this flow if you do not want to fix the relative placement groups after you run the `create_fp_placement` command.

[Figure 5-20 on page 5-42](#) shows the default relative placement flow.

Figure 5-20 Default Relative Placement Flow



Use the `legalize_fp_placement` command to automatically legalize the standard cells. The standard cells in relative placement groups may be moved outside of the corresponding relative placement bounding box during legalization.

Note:

The `legalize_fp_placement` command does not currently support relative placement groups.

Using the Relative Placement Cells Flow

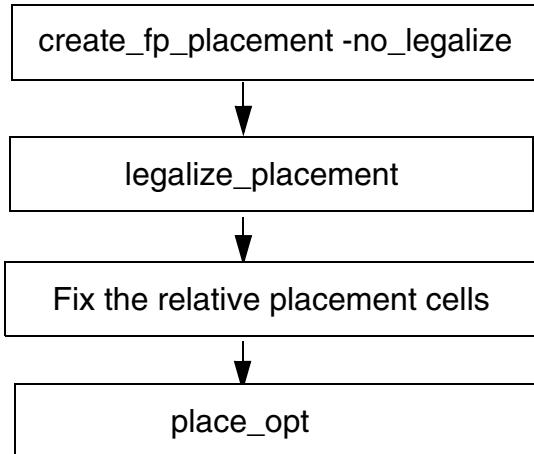
Use this flow if your design contains either macro-only or standard cell-only relative placement groups and you need to fix the relative placement groups after floorplanning to ensure the exact relative placement constraints are met.

1. Define the relative placement constraints.
 - Create the relative placement groups by using the `create_rp_group` command.
 - Add relative placement objects to the groups by using the `add_to_rp_group` command.
2. Run virtual flat placement by using the `create_fp_placement -no_legalize` command.
3. Mark all macros as `is_fixed`.
4. Legalize your design by using the `legalize_placement` command.
5. Mark all relative placement cells as `is_fixed`.
6. Perform physical placement by using the `place_opt` command.

You can skip steps 4 and 5, depending on your design requirements.

[Figure 5-21 on page 5-43](#) shows the flow for fixing the relative placement cells.

Figure 5-21 Fixing the Relative Placement Cells Flow



Placing Multiply Instantiated Modules

You can perform virtual flat placement of multiply instantiated modules. Multiply instantiated modules are physically represented in the design as plan groups and have the same reference. A set of these instantiated modules is called a multiply instantiated module group. A design can have more than one multiply instantiated module group. If for example, modules A1 and A2 have reference A, and modules B1 and B2 have reference B, two multiply instantiated module groups are formed. Many designs use multiple instances of the same logic block to create the required functionality. By using the same block for similar operations, the design time is reduced.

This section includes the following topics:

- [Determining Which Plan Groups and Soft Macros are Multiply Instantiated Modules](#)
- [Identifying Multiply Instantiated Module Plan Groups](#)
- [Shaping the Multiply Instantiated Module Plan Groups](#)
- [Placing and Analyzing the Multiply Instantiated Module Plan Groups](#)
- [Flipping the Placement of Multiply Instantiated Module Plan Groups](#)
- [Performing a QoR Analysis of Multiply Instantiated Module Plan Groups](#)

Determining Which Plan Groups and Soft Macros are Multiply Instantiated Modules

Before you can place the multiply instantiated modules, you must first determine which plan groups and soft macro are to be considered as multiply instantiated modules. This can be done by using the `uniquify_fp_mw_cel -store_mim_property` command to store the multiply instantiated module information in the Milkyway database as a multiply instantiated module property. After uniquification, this makes it possible for other IC Compiler design planning commands to access the multiply instantiated module information and treat the cell instances with these properties as multiply instantiated modules in the virtual flat placement design planning flow.

The `uniquify_fp_mw_cell` command uses the following syntax:

```
uniquify_fp_mw_cel -store_mim_property cell_instances
```

For example:

```
uniquify_fp_mw_cell -store_mim_property [get_cells {instA instB instC}]
```

If you select the `-store_mim_property` option, the tool stores a multiply instantiated module property on the cell instances you specify during uniquification. Specifically, it stores the name of the original unique master for every specified hierarchical cell instance as a multiply instantiated module property.

Removing the Multiply Instantiated Module Property

You can use the `remove_mim_property` command to remove the multiply instantiated module data for selected cell instances. You must specify a list of hierarchical cells for which the multiply instantiated module data is to be removed. If a cell is not found, a warning message is issued.

The cell instances are no longer treated as multiply instantiated modules; each cell instance is now referred to as a unique master.

The `remove_mim_property` uses the following syntax:

```
remove_mim_property collection_of_cells
```

For example:

```
remove_mim_property [get_cells instA]
```

Identifying Multiply Instantiated Module Plan Groups

Use the `report_mim` command to print a list of all properly identified multiply instantiated module plan groups in the design. Soft macro multiply instantiated modules are also identified by this command, although for placement, they will be treated as hard macros.

Here is an example of a report:

```
icc_shell > report_mim
*****
Report      :MIM
Design      :test.CEL;1
Version     :B-2008.09-ICC-CS
Date        :Mon Aug 25 17:01:51 2008
*****
Master BLENDER is instantiated to:
  Soft Macro: 1_ORCA_TOP/1_BLENDER_4
  Soft Macro: 1_ORCA_TOP/1_BLENDER_3
  Soft Macro: 1_ORCA_TOP/1_BLENDER_2
  Soft Macro: 1_ORCA_TOP/1_BLENDER_1
  Soft Macro: 1_ORCA_TOP/1_BLENDER_5
  Reference Blender is instantiated 5 times.
Total 1 MIM references and 5 MIM instances.
MIM plan group flipping grid:x offset=425.5,x step=0.410
                           y offset=425.565,y step=7.380
MIM Report End
```

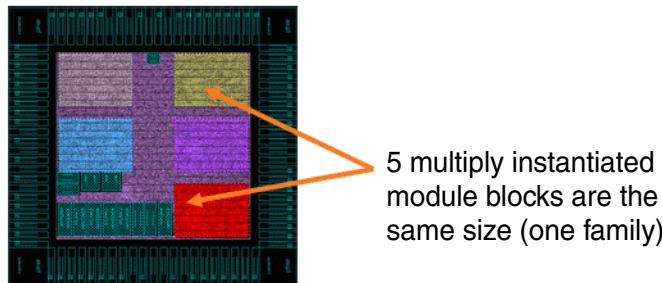
The report also provides information about the status of the copied placement data and suggests a multiply instantiated module plan group flipping grid on which to place an multiply instantiated module plan group. When the plan group is flipped, the standard cell placement remains legal.

Shaping the Multiply Instantiated Module Plan Groups

After running virtual flat placement (`create_fp_placement` command), you can use the `shape_fp_blocks` command to shape the plan groups in each multiply instantiated module group automatically. (See “[Automatically Placing and Shaping Objects In a Design Core](#)” in [Chapter 6](#).)

The plan groups in each multiply instantiated module group are shaped and sized identically, as shown in [Figure 5-22 on page 5-46](#).

Figure 5-22 Plan Groups of Multiply Instantiated Module Groups in the Same Size and Shape



All corners of the multiply instantiated module plan groups are aligned on their row/column placement boundaries and are placed in such away that they can be legally flipped, provided the rows are uniform.

Note:

The default plan group locations might not be aligned.

Placing and Analyzing the Multiply Instantiated Module Plan Groups

Run the `create_fp_placement` command to place the macros and standard cells into fixed placement plan group boundaries. However, the placement in the multiply instantiated module groups will not be identical.

You can use the `copy_mim` command and options to perform various manipulations during the virtual flat placement stage on the multiply instantiated modules to determine which of the plan groups in each multiply instantiated module group is the master of that multiply instantiated module group.

The placement of the master is copied to the other plan groups in the multiply instantiated module group. The copied information, which is saved to the Milkyway database, contains the original placement of all the cells in the copied plan multiply instantiated module plan groups relative to the lower-left corner of the plan group and the orientation of the plan group.

The `copy_mim` command uses the following syntax:

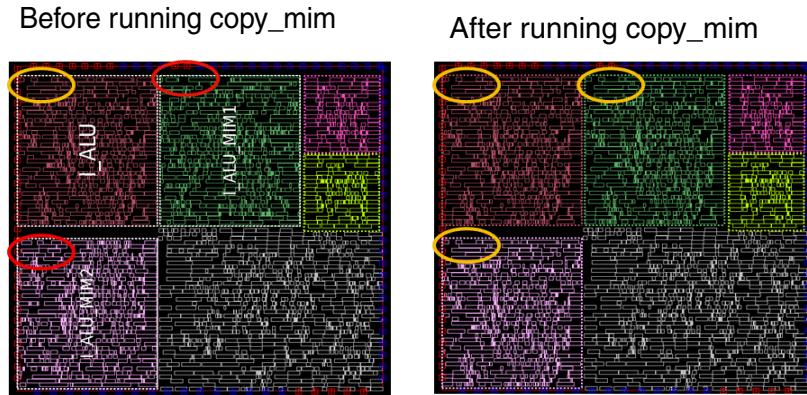
```
copy_mim
[-type placement | blockage | boundary]
[-restore_placement]
collection
```

A *collection* specifies a list containing one multiply instantiated module plan group.

As shown in [Figure 5-23](#), the placement is different in every plan group before the `copy_mim` command is run. After the `copy_mim` command is run, the placement argument copies the cell placement of the given multiply instantiated module to the other multiply instantiated modules in the same group, as shown in the following example:

```
icc_shell> copy_mim -type placement 1_ALU
```

Figure 5-23 Before and After Running the copy_mim Command



You can create multiply instantiated modules with the same cell placement, blockages, and shapes as described in the following sections.

Copying the Cell Placement of a Plan Group

Use the `copy_mim -type placement` option to copy the cell placement of a plan group specified as a *collection* to all other plan groups in its multiply instantiated module group where the plan group in the list is the master. The copied information contains the original placement of all the cells in the copied multiply instantiated module plan groups relative to the lower-left corner of the plan group and the orientation of the plan group. This is the default.

Note:

Placement blockages are not touched by this command. Errors are reported if the *collection* is not part of an multiply instantiated module group or the plan groups in the plan group's multiply instantiated module group do not have the same size and shape.

You can make more than one copy on an multiply instantiated module group; however, you will lose a previous copy because only one level of undo is supported.

You can print out information about which multiply instantiated module groups are copied and the master plan group for that copy by using the `report_mim` command. If the shapes or sizes of the plan groups changed from the previous `copy_mim -type placement` command, this information is also reported.

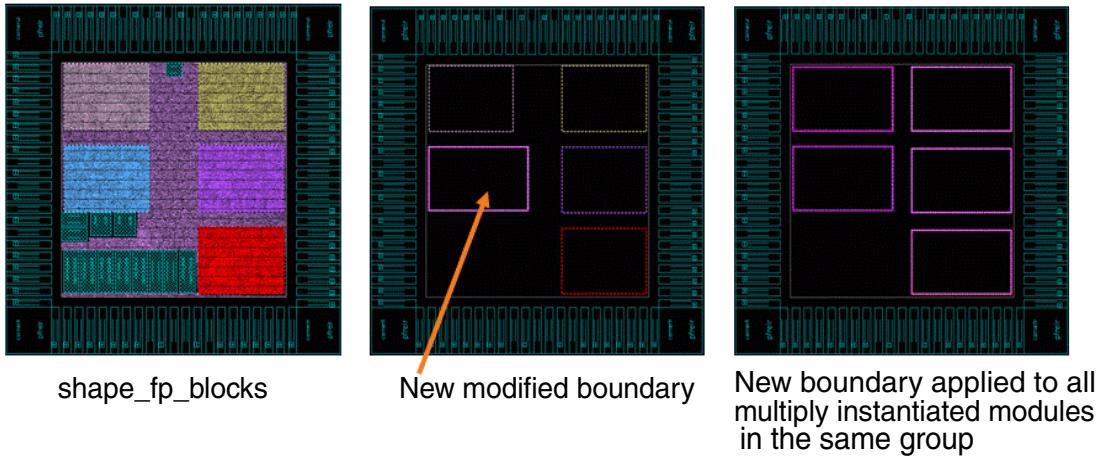
Copying Placement Blockages and Boundaries to Other Plan Groups

Use the `copy_mim -type blockage` option to copy the placement blockages that intersect the plan group's specified `collection` to all the other plan groups in the `collection` multiply instantiated module group. This command uses the plan group orientation you set with the `flip_mim` command. (See “[Flipping the Placement of Multiply Instantiated Module Plan Groups” on page 5-49.](#)”)

Use the `copy_mim -type boundary` option to reshape each plan group that belongs to a multiply instantiated module group so that their boundaries are identical to the specified plan group in the `collection`, while at the same time preserving the orientation of the individual plan groups.

[Figure 5-24](#) shows an example of a new modified boundary applied to all the other multiply instantiated modules in the same group.

Figure 5-24 Modified Boundary Applied to Other Multiply Instantiated Modules in the Same Group



Restoring the Placement of Cells in Plan Groups

Use the `copy_mim -restore_placement` option to restore the placement of cells in all the plan groups specified in the multiply instantiated module group as it was when the last `copy_mim -type placement` was done.

The orientation or flipping about the x- or y-axis of `PG_list` is also restored to what it was during the `copy_mim -type placement` command.

Note:

Placement blockages are not touched by this command. Errors are reported if any size or shape does not correspond to the original size or shape in the `PG_list` multiply instantiated module group.

Flipping the Placement of Multiply Instantiated Module Plan Groups

Use the `flip_mim` command to flip the placement of all cells and their shapes in the multiply instantiated module plan group *collection* list and set the orientation. The resulting pins on the multiply instantiated module are also flipped. After running the `flip_mim` command, the resulting placement is the same as if the plan group was converted to a soft macro, flipped, and then converted back to a plan group.

The `flip_mim` command uses the following syntax:

```
flip_mim  
[-direction x | y]  
collection
```

Horizontal Requirements for Plan Group Locations

The following horizontal requirements for plan group locations when running the `flip_mim` command ensures that the cells in a plan group have the exact same relative distance to the left and right plan group boundaries after flipping the plan group horizontally.

- Snap the left and right boundaries of a plan group to the edge of a unitTile or to a location that is one-half the unitTile width.
- Ensure that the plan group width is a multiple of the unitTile width.

Vertical Requirements for Plan Group Locations

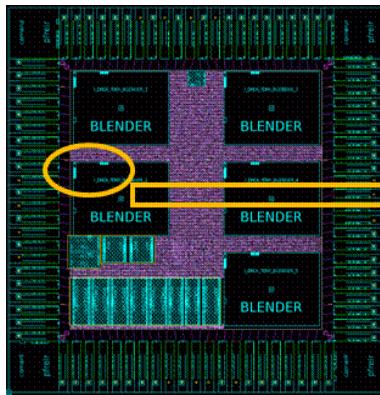
The following vertical requirements for plan group locations when running the `flip_mim` command ensures that the cells in a plan group have the exact same relative distance to the top and bottom plan group boundaries after flipping the plan group vertically. The cells are flipped and placed legally at the rows with the opposite orientation.

- Snap the top boundary of a plan group to the top-side of a row and snap the bottom boundary of a plan group to the bottom-side of a row. Alternatively, you can snap both the top and bottom boundaries of the plan group to a location that is one-half the row height
- Ensure that plan group height covers an even number of rows. (It must not be an odd number of rows.)

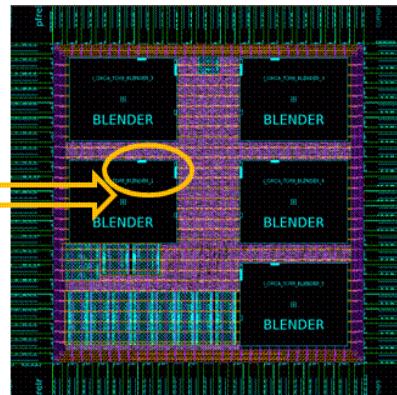
[Figure 5-25 on page 5-50](#) shows a before and after example of running the `flip_mim` command.

Figure 5-25 Before and After Running the flip_mim Command

Before running flip_mim



After running flip_mim



Only mirroring, flipping about the x- or y-axis, is allowed for multiply instantiated module plan groups. If the *collection* list is not part of an multiply instantiated module group and if the plan group is not rectangular, an error is reported.

During flipping, the bounding box of the plan group does not move.

Placement blockages are not flipped; however, the `copy_mim -type blockage` command recognizes the flipped orientation.

Using the Multiple Instantiated Module GUI

You can run multiple instantiated module operations from a single dialog box by choosing Floorplan > Multiple Instantiated Modules from the GUI menu. The MIM Flow dialog box appears.

Performing a QoR Analysis of Multiply Instantiated Module Plan Groups

Use the `get_fp_wirelength` command to run a quality of results (QoR) analysis of the different multiply instantiated module plan groups.

The following options help determine which plan group out of a set of multiply instantiated module plan groups has the best placement with respect to timing.

- Plan Group Interface Nets – Reports the wire length of interface nets for all plan groups. Interface nets have at least one pin inside the plan group and one pin outside the plan group boundary.
- Plan Group Internal Nets – Reports the wire length of internal nets for all plan groups. Internal nets have all their pins inside the plan groups.

Generating a QoR Placement Report

You can generate a quality of results (QoR) report for virtual flat placement by using the `report_fp_placement` command. It includes the following information:

- Total wire length
- Number of regions with high density
- Number of regions with low density
- Number of overlaps between hard macros
- Number of overlaps between hard macros and standard cells
- Number of cells that are not inside their respective plan groups (or core area)
- Number of overlaps between plan groups
- Number of cells violating the core area

If your design does not have channels, you can use the `-check_abutment` option to check for the abutment of each soft and hard macro with the surrounding macro cell instances. The abutment check reports gaps detected between the macro cell instances.

Checking the Total Wire Length Estimate

You can use the `get_fp_wirelength` command to evaluate the over all placement quality. The command generates vertical, horizontal, and total wire length statistics for the top level signal nets in your design. It also calculates the virtual route wire length as a minimum length needed to connect all the pins.

Performing Floorplan Editing

You can use IC Compiler to perform the following floorplan editing operations:

- Creating objects
- Deleting objects
- Undoing and redoing edit changes
- Moving objects
- Changing the way objects snap to a grid
- Aligning movable objects

- Editing rectilinear objects
 - Turning snapping on or off
 - Cutting and adding object shapes
 - Changing object shapes to rectangle, T-shape, L-shape, cross, or U-shape
 - Moving and resizing objects
- Resizing objects manually
- Expanding objects
- Distributing objects
- Spreading objects
- Splitting objects
- Fixing and unfixing objects for edits

For detailed information, see the “Editing a Design” topic in IC Compiler online Help.

Using Constraints With the Core and Die Editing Commands

You can use the “keep placement” and Keep pad-to-core distance” constraints with the editing commands that are relative to the core and die. Using these constraints makes it easier and more convenient to refine and debug core and die commands.

The constraints are effective only when they are applied to the following core and die editing commands:

```
set_object_boundary  
resize_objects  
set_object_shape  
window_stretch  
move_objects  
cut_objects
```

When you move or change a core or die, you can use the `-keep_placement` option to ensure that standard cells and hard macros so that they remain within the boundary of the parent object (for example, the core, the die, or a plan group).

You can use the `keep_pad_to_core_distance` constraint to maintain the distance between the core and the I/O pad cells and by implication, the distance between the core and the die. This forces changes to the core and die to be made in tandem. If the I/O pad cells are absent, the core-to-die distance is still maintained.

Note:

After moving or resizing objects, the resulting placement is not legalized. You must explicitly legalize the placement by using the `legalize_placement` command.

Enabling the Feasibility Flow for Optimizing Dirty Constraint Data

You can perform a feasibility analysis during preroute optimization on designs with preliminary netlists and dirty constraint data in the early stages of the design planning flow. You can perform limited optimization on the parts of the design with incomplete constraint data (dirty netlists), and complete optimization on the parts of the design that are well constrained. By performing limited optimization on the dirty constrained part of the design, you can quickly identify potential design problems early in the design planning stage and get information on the overall timing feasibility of the design.

This section includes the following topics:

- [Recommended Feasibility Optimization Flow](#)
- [Creating a Collection of Pins](#)
- [Reporting the Feasibility Options](#)

Use the `set_feasibility_options -enable true` command to enable the feasibility mode for the `place_opt` command and the reporting commands (`report_qor`, `report_timing`, `report_constraint`, and `get_timing_paths`). The feasibility mode identifies the timing feasibility of the design. When set to `true`, it relaxes the constraints by filtering out impossible to meet timing paths for reporting commands such as `report_qor` and `report_constraint`. (Note that endpoints with zero path violations and zero wire load violations are considered unachievable endpoints.) Then, instead of spending time optimizing unachievable timing paths, you can flag these unfixable paths and quickly move on and optimize the other timing paths in the design.

The `set_feasibility_options` command uses the following syntax:

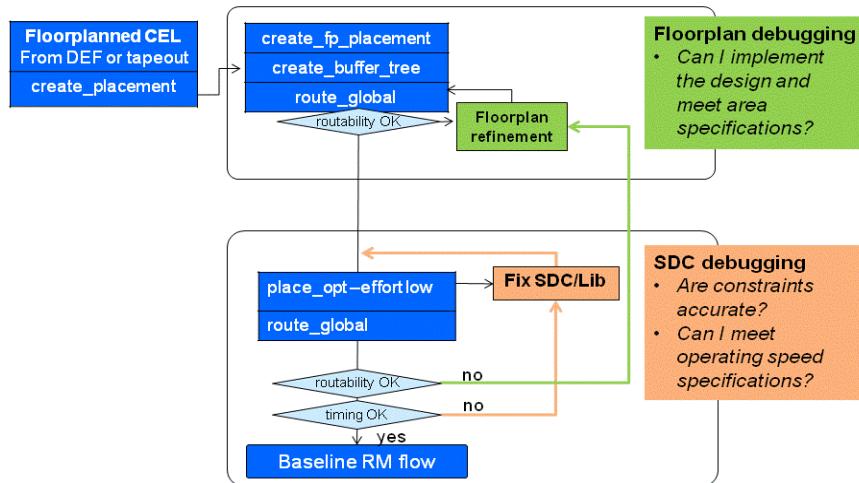
```
set_feasibility_options [-enable true | false]
[-zero_path_margin zero_path_margin_num]
[-zero_wire_load_margin zero_wire_load_margin_num]
[-io_margin io_margin_num]
[-group_io boolean-string]
[-verbose_level verbose_level_num]
[-feasibility_reporting boolean-string]
[-zero_path_violations boolean-string]
[-zero_wire_load_violations boolean-string]
```

The `place_opt -effort low` command applies feasibility mode constraint relaxation before starting preroute optimization. You can use this command during the feasibility analysis flow to improve the runtime and quality of results (QoR) of the placement.

Recommended Feasibility Optimization Flow

The recommended feasibility optimization flow is shown in [Figure 5-26](#).

Figure 5-26 Feasibility Optimization Flow for Optimizing Dirty Netlist Constraints



Follow these steps to run the feasibility optimization flow.

1. Perform an initial virtual flat placement of the hard macros and standard cells in the design by using the `create_fp_placement` command.
2. Search the design to locate automatic high-fanout synthesis (AHFS) nets and create buffer trees for them by using the `create_buffer_tree` command. The command creates a buffer tree for each specified driver pin and for the driver pin of each specified net.
3. Perform a quick global route of the design by using the `route_fp_proto` command. Check the congestion. If it is not acceptable, go back and refine the floorplan.
4. Enable the feasibility mode for the `place_opt` and reporting commands by using the `set_feasibility_options -enable true` command. The feasibility mode identifies the timing feasibility of the design. When set to `true`, it relaxes the constraints by filtering out unachievable and impossible to meet timing paths for reporting commands such as `report_qor` and `report_constraint`.

You can get a report of all the options set for feasibility optimization by using the `report_feasibility_options` command.

5. Perform limited optimization on the parts of the design with incomplete constraint data (dirty netlists), and complete optimization on the parts of the design that are well constrained by using the `place_opt -effort low` command. This command also prints a summary of the feasibility timing adjustments performed per path group.

Note:

If high-fanout synthesis (HFS) has already been run on the design, this command uses the existing high-fanout synthesis tree.

6. Use the reporting commands to report the feasibility quality of results (QoR) and timing in the feasibility mode. The following reporting commands will consider the adjusted timing constraints at endpoints while reporting the results in feasibility mode.

```
report_qor
report_timing
report_constraint
get_timing_paths
```

You can use the `report_qor` and `report_constraint` commands to report the original timing constraints and adjusted timing constraints in the same report.

You can use the `report_timing` command to report paths based on the original timing constraints.

Note:

If the endpoint reported by the `report_timing` command has an adjustment margin specified, then the `report_timing` command will also show the adjusted slack.

7. Disable the feasibility mode by using the `set_feasibility_options -enable false` command. The constraint relaxation feature is disabled and the adjusted required times are reset.
8. Perform a global route on the design by using the `route_global` command. Check the congestion before proceeding to detailed implementation.

Creating a Collection of Pins

During feasibility optimization, you can use the `get_adjusted_endpoints` command to get the pins (endpoints) with the adjusted required time (margin values) that match the specified criteria and scenarios.

The `get_adjusted_endpoints` command uses the following syntax:

```
get_adjusted_endpoints
[-zero_path | -zero_wire_load | -io | -all]
```

Reporting the Feasibility Options

You can get a report on all the options that were set during feasibility optimization by using the `report_feasibility_options` command.

6

Creating and Shaping Plan Groups

This chapter describes how to create plan groups for logic modules that need to be physically implemented as a group. Plan groups restrict the placement of cells to a specific region of the core area. This section also describes how to automatically place and shape objects in a design core, add padding around plan group boundaries, and prevent signal leakage and maintain signal integrity by adding modular block shielding to plan groups and soft macros.

This chapter includes the following sections:

- [Creating Plan Groups](#)
- [Analyzing and Manipulating the Hierarchy](#)
- [Automatically Placing and Shaping Objects In a Design Core](#)
- [Enabling Plan Group-Aware Scan Chain Reordering](#)
- [Adding Padding to Plan Groups](#)
- [Adding Block Shielding to Plan Groups or Soft Macros](#)

Creating Plan Groups

You create plan groups to manipulate the hierarchy. (See “[Analyzing and Manipulating the Hierarchy](#)” on page 6-4.) A plan group is a physical partition with a physical boundary. Each plan group represents a module in the logic hierarchy that you want to implement as a physical block and contains cells that belong to that module. A plan group can have many logical modules. The physical block inherits the size and shape (rectangular or rectilinear) of the plan group. Timing budgeting, pin cutting, and commit hierarchy are applied on plan groups.

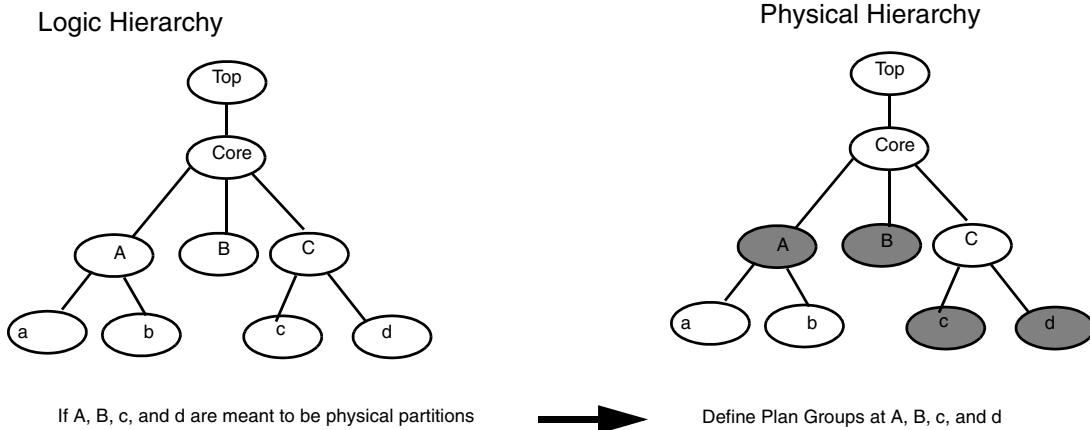
Plan group boundaries are created for the modules you selected and are placed outside the chip boundary. Each plan group boundary is displayed in a different color in the layout window and in the hierarchy tree to illustrate the relationship between the logical hierarchy and the physical groups of standard cells.

Plan groups require all their cells to be placed inside them and prohibit the placement of other cells in the same area.

In the early stage of the design flow when your design contains the intended plan group partitions, you can analyze the pins and feedthrough pins to help identify issues before you create the pins and fix the pin locations. Using this information, you can redefine route guides or constraints on the router to achieve better pin locations.

During commit hierarchy (`commit_fp_plan_groups` command), the plan groups are converted to soft macro cells, also called a child cell, or simply a physical block, and the pins on each soft macro are cut. (See “[Converting Plan Groups to Soft Macros](#)” in Chapter 13.) After the conversion is complete, the floorplan will contain only the top-level standard cells and soft macro cells. Place and route can then work on each soft macro independently. These soft macros are then propagated to the top-level in a FRAM view, and place and route is run on the whole chip design.

You can create plan groups for the logic modules that exist at any level in the logic hierarchy. [Figure 6-1 on page 6-3](#) shows the relationship between the logic hierarchy and the physical hierarchy.

Figure 6-1 Relationship Between Logic Hierarchy and Physical Hierarchy

To create a plan group,

1. Choose Floorplan > Create Plan Group.

The Create Plan Group dialog box appears.

Alternatively, you can use the `create_plan_groups` command.

2. Click in the “Hierarchical cell” text box and enter the hierarchical cell names. A plan group is created for each valid hierarchical cell name in the list.
3. Define the plan group shape by doing one of the following:

- Select the Aspect option to define a plan group shape by its aspect ratio and utilization. This is the default.

Enter an aspect ratio (height divided by width) for the plan group area. If you specify a ratio of 1.00 (the default), the height and width are the same, and therefore the core is a square. If you specify a ratio of 3.00, for example, the height is three times the width.

Enter a utilization value. This number is calculated as the total plan group area divided by the area occupied by the cells of the plan group.

- Select the Dimension option to define a plan group shape by its dimensions and enter the width and height in microns. The plan groups are placed outside the core area.
- Select the Region option to define a plan group by using coordinates.

To define a rectangular plan group shape, select the Rectangle button and draw the rectangle in the layout view, or enter the lower-left and upper-right coordinates in the Coordinates field.

To define a rectilinear plan group shape, select the Rectilinear button and draw the rectilinear shape in the layout view, or enter the lower-left and upper-right coordinates in the Coordinates field.

4. (Optional) You can apply a specific color to a plan group. To do this, deselect the “Cycles colors” option to assign a new color to each new plan group automatically (the default). Select the “Specified” option to specify a particular color for the plan group color from the list. Select “None” for no color

By default, IC Compiler automatically assigns a color to the plan group.
5. Click OK or Apply.

Removing the Plan Groups

To remove (delete) plan groups from the current design, choose Floorplan > Delete Floorplan Objects > Delete All Plan Groups. This command lets you selectively delete one or more plan groups.

Alternatively, you can use the `remove_plan_groups` command.

Note:

Deleting a plan group means cancelling the grouping for future placement. The existing placed cells are not affected.

Analyzing and Manipulating the Hierarchy

You can use the hierarchy browser to navigate through the design hierarchy and to examine the logic design hierarchy and display information about the hierarchical cells and logic blocks in your design. You can select the hierarchical cells, leaf cells, or other objects you want to examine in layout or schematic views.

If you are not familiar with a design, you can explore the logic hierarchy to understand its structure and gather information about the design objects. You can also use the hierarchical browser to

- Highlight cells or blocks (and their leaf cells) with different colors in both the hierarchy browser and the layout window
- Perform floorplan tasks such as creating plan groups and estimating black boxes
- Display design schematics of hierarchical cells
- View the hierarchical data interactively and switch between the layout view and the hierarchy browser view for selection and highlighting
- Perform probing between the logical hierarchy tree and the physical layout. This allows you to select cell instances, highlight and unhighlight cells, turn flylines on and off, and display the connectivity of selected modules and plan groups.

This section includes the following topics:

- [Opening the Hierarchy Browser](#)
 - [Exploring the Hierarchical Structure](#)
 - [Manipulating the Hierarchy](#)
-

Opening the Hierarchy Browser

To open the hierarchy browser, do one of the following:

- In the layout window, choose Partition > New Hierarchy Browser View
- In the main window, choose Hierarchy > New Hierarchy Browser View

The view window consists of two panes with an instance tree on the left and an object table on the right. The instance name of the top level design appears at the top of the instance tree.

Note:

Once the instance tree is loaded, it exists in memory. If you close the hierarchical browser, and then open it later, the existing instance tree is loaded automatically, which saves you a significant amount of time, particularly if you have a multimillion cell design.

Indicating Object Types

The hierarchy browser indicates object types by displaying icons next to the cell names.

- Logical
 - T – Top level design
 - BB – Black boxes
 - H – Hierarchical module
 - HM – Hard macro
 - IO – I/O cell
 - SC – Standard cell
 - ILM – Interface logic model
- Physical
 - P – Plan group
 - SM – Soft macro

If you highlight cell hierarchies with colors, the hierarchy browser displays the cell colors on the icons.

Exploring the Hierarchical Structure

You can explore the complete hierarchical structure of your design and observe how many hierarchical blocks are present. To explore the design hierarchy, you can

- Click the expansion button (plus sign) next to an instance name to expand the instance tree, showing the names of the subblocks at the next level of hierarchy.

You can also right-click and select “Expand Tree Level” from the menu to display the Expand Tree dialog box. You can expand and display the instance tree up to a user-specified level; the root is a level 1, which shows the blocks one level below the current level. A number greater than 1 must be specified in the “Expand to level” field to expand the instance tree and see the subblocks at the next level.

- Select an instance to display leaf cell information in the object table.

Shift-click or Control-click instance names to select combinations of instances. The information includes cell instance names, cell types, and cell reference names.

- Display port and pin names or net names by selecting a port, pin, or net (rather than a cell) in the field above the object table list on the right.

You can specify what types of information are displayed in the object table list. Right-click inside the table and select Columns, and select the items you want displayed. You can display the design statistics for the object types in the list. This includes the cell reference names, the number of soft or hard macros in a particular module, the utilization area, the number of standard cells, the number of pins, and the absolute area.

You can also right-click and select “Columns.” Select “More...” from the menu to display the “Item Select” dialog box. You can then select the details you want to display from the objects listed in the dialog box.

Note:

For more information about the hierarchy browser, see IC Compiler online Help.

Manipulating the Hierarchy

This section describes how to create a new level of hierarchy in the early design planning or prototyping phase and how to remove a level of hierarchy from the current design.

This section includes the following topics:

- [Merging the Hierarchy](#)

- [Flattening the Hierarchy](#)

Merging the Hierarchy

You can select any number of cells in the current design and merge (group) them together into a new cell, thereby creating a new level of logic hierarchy.

To create a new level of hierarchy,

1. Select Partition > Merge Hierarchy.

The Merge Hierarchy dialog box appears.

Alternatively, you can use the `merge_fp_hierarchy` command.

2. Set the options, depending on your requirements.

- New cell name – Enter the name of the new cell instance that replaces the merged cells in the design.
- New design name – Enter the reference name of the new cell design created by merging the existing cells.

The name you enter cannot already exist in the current design.

- Cells – Enter a list of cells (modules, plan groups, black boxes, hard macros, soft macros, or leaf cell instances) that you want to merge together into a new level of logic hierarchy.

If you specify more than one cell, you must enclose them in braces { }.

The cells you select must be from the same parent and reside at the same level of hierarchy.

- Update SDC – During the creation of the logical module, you may want to update your original Synopsys Design Constraints (SDC) file so that it matches the new logical hierarchy. Select this option to update the SDC file.

Click in the “Input SDC file” box and enter the name of the original SDC file to be updated to reflect the netlist changes. The file should contain valid SDC commands for the current design.

Click in the “Output SDC file” box and enter the name of the resulting SDC output file. This file should contain the same information as the original SDC file but with updated top-level timing constraints and new hierarchical names for all the objects in the netlist.

- Load back updated SDC file – Select this option to reload the updated SDC file. The new SDC file is automatically reloaded into the CEL view.

- Create wrapper – Select this option to create a wrapper on any existing leaf cell or hierarchical logic modules. A wrapper is used to selectively expose internal connections and dangling pins onto a wrapper interface so that any future changes made to these internal objects do not impact the wrapper interface.

The new wrapper module retains the pin names from the logic modules. (Note that if the new pin name on the wrapper is different from the one connected inside the original cell's pins, a file describing the mapping of the pin names is automatically generated.)

You can also use this option to create a wrapper module for a merged plan group before committing it to a soft macro.

Note:

Pin assignment and feedthrough generation occur only at the wrapper module; the original module under the wrapper will not be affected by any feedthrough insertion changes.

If there are changes in the netlist of the original logic modules, but the module interface is still the same, you can swap the new netlist into the wrapper module, create a new block CEL view, and then link it to the top level design. You do not have to run pin assignment or generate feedthroughs again for the new netlist.

- Port Merging – Select this option if you do not want the ports connected by the same interface net to be merged under the wrapper.

3. Click OK or Apply.

The Milkyway hierarchical netlist is updated and a new hierarchical cell is created in the Hierarchical Preservation data.

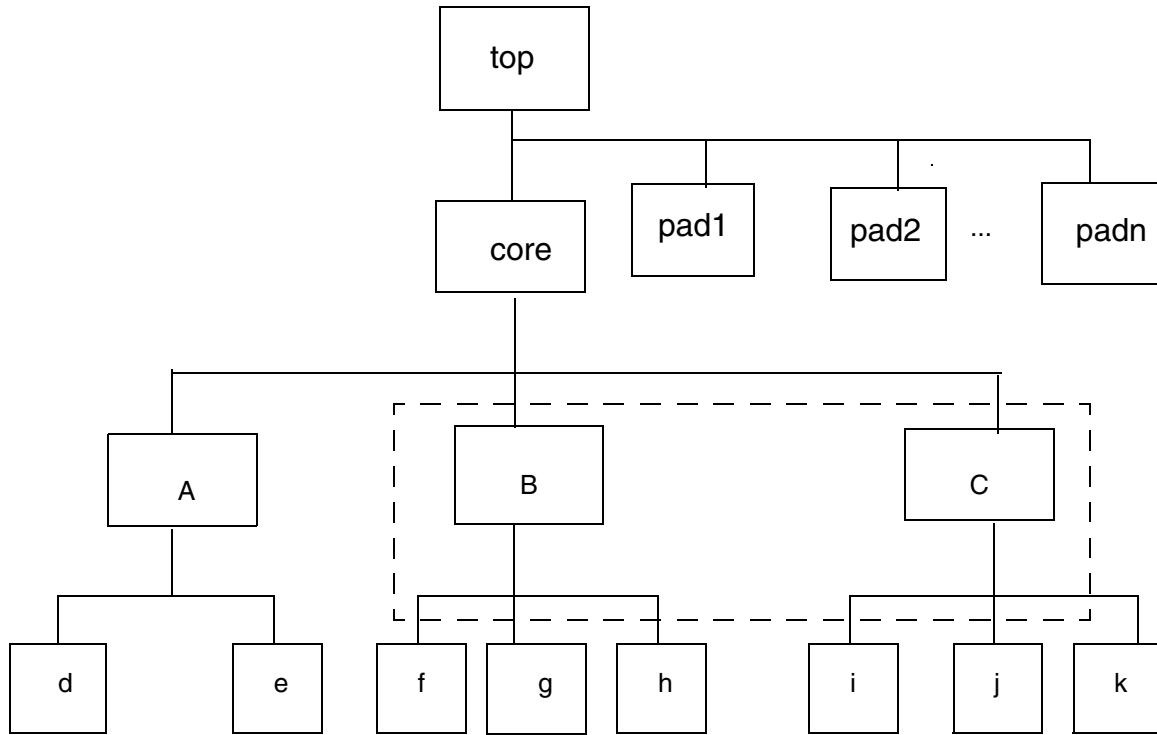
This new logical module can be converted into plan groups, which you can then later commit (`commit_fp_plan_groups`) to soft macros (physical blocks).

Logic Netlist Grouping (Merging) Examples

You can group (merge) modules at the same level of hierarchy and of the same parent in the logic hierarchy. This section has five examples, showing ways that modules can and cannot be grouped.

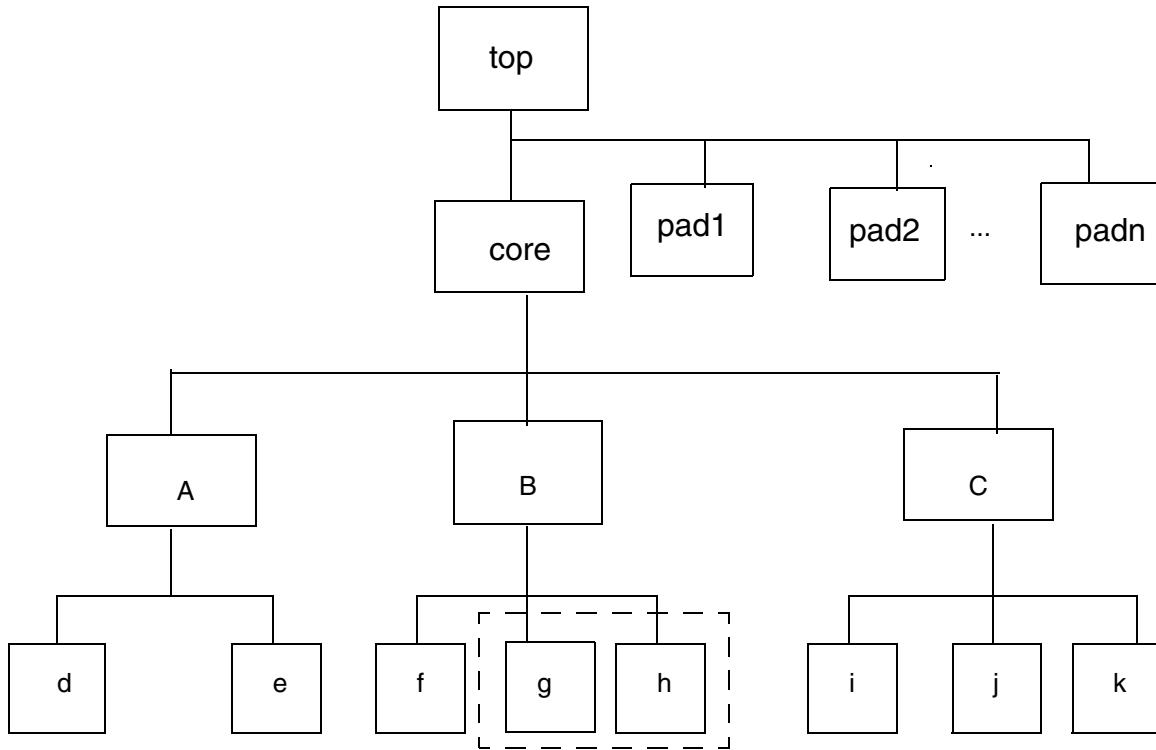
In example 1, shown as [Figure 6-2 on page 6-9](#), standard cells at the same level as A, B, and C are included in the merge.

Figure 6-2 Example 1: Logic Netlist Merging: Same Level of Hierarchy and Same Parent



In example 2, shown as [Figure 6-3 on page 6-10](#), standard cells at the same level as f, g, and h and under B, are included in the merge. Standard cells and hard macros of B and f, are placed at the top level.

Figure 6-3 Example 2: Logic Netlist Merging: Same Level of Hierarchy and Same Parent



The remaining three examples ([Figure 6-4](#), [Figure 6-5](#), and [Figure 6-6](#)) show that you cannot merge modules that are in multiple levels of the hierarchy or that do not have the same parent.

Figure 6-4 Example 3: Logic Netlist Merging: Multiple Levels of Hierarchy (Not Possible)

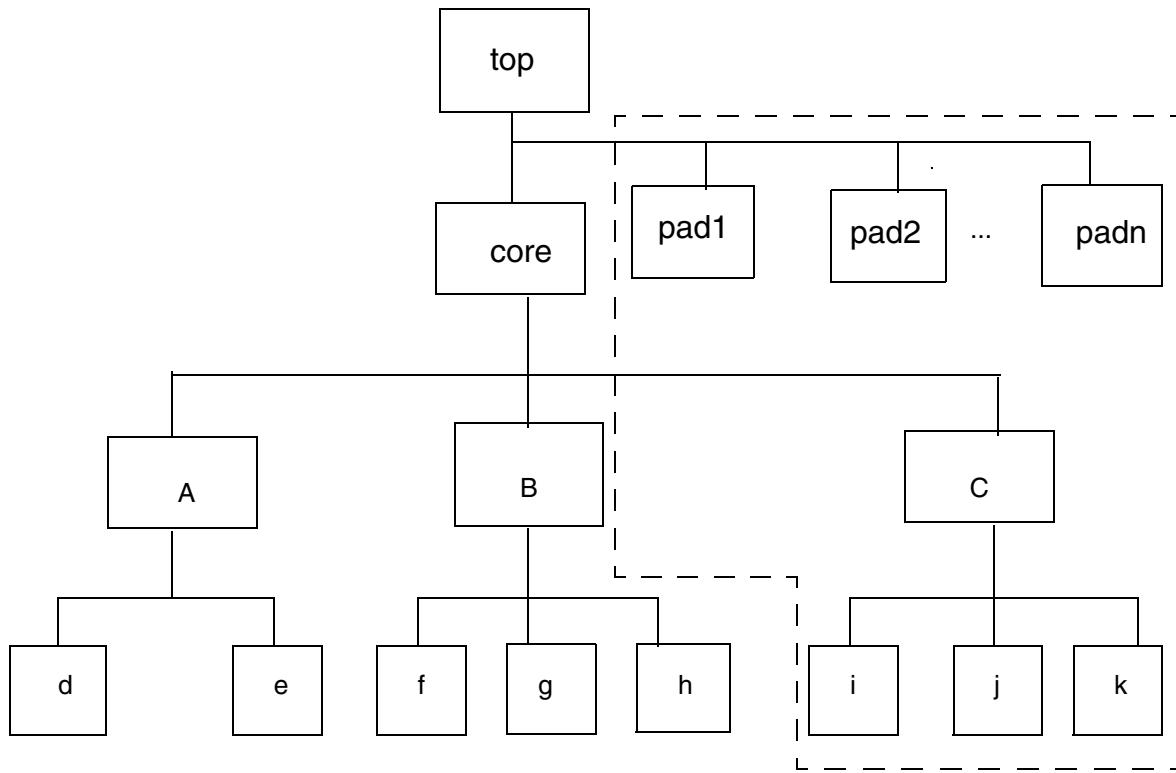


Figure 6-5 Example 4: Logic Netlist Merging: Multiple Levels of Hierarchy and Not the Same Parent (Not Possible)

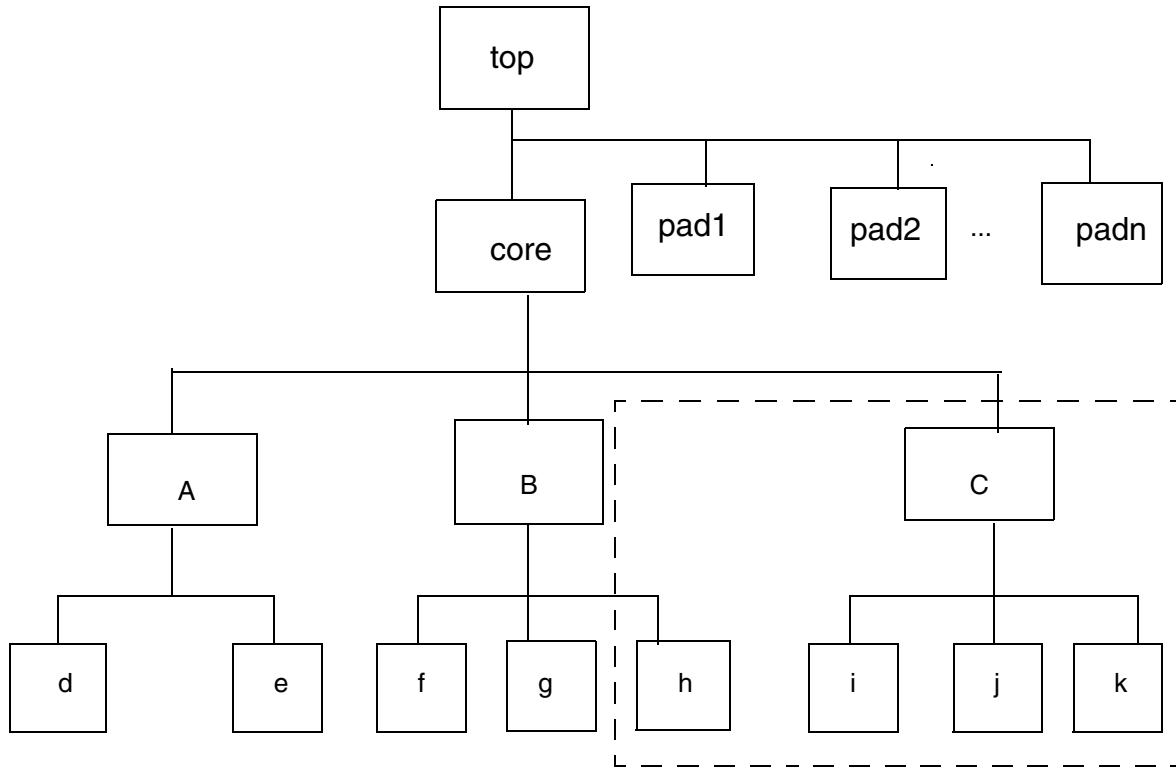
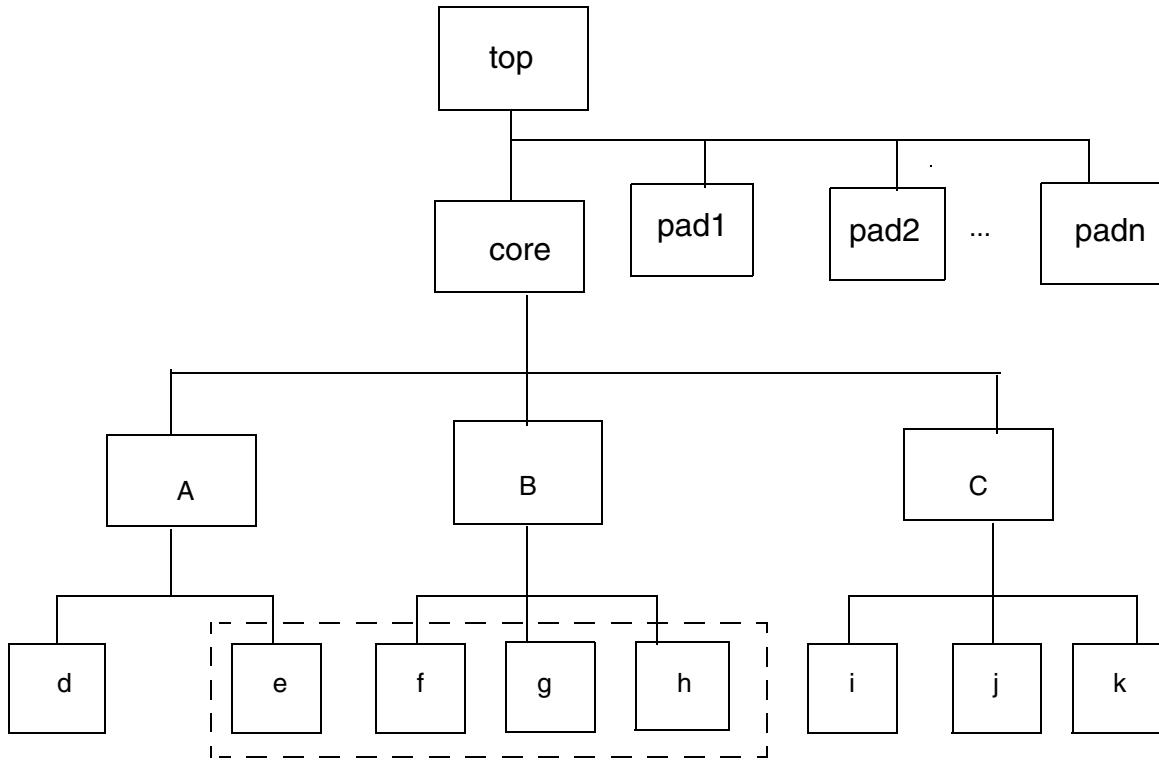


Figure 6-6 Example 5: Logic Netlist Merging: Not the Same Parent (Not Possible)



Flattening the Hierarchy

You can flatten (remove) a single level of hierarchy from your current design so that the logic modules at the next lower level of hierarchy are merged into the current level of hierarchy. This replaces a single hierarchical block with multiple blocks brought up from the lower level, which raises all of those blocks up by one level of hierarchy.

To flatten a level of hierarchy,

1. Select Partition > Flatten Hierarchy.

The Flatten Hierarchy dialog box appears.

Alternatively, you can use the `flatten_fp_hierarchy` command.

2. Set the options, depending on your requirements.

- Cells – Enter a list of cells (modules, plan groups, black boxes, hard macros, soft macros, or leaf cell instances) in the current design that are to be flattened

If you specify more than one cell, you must enclose them in braces { }.

- Update SDC – Select this option to update the SDC file to reflect the changes to the netlist.

Click in the “Input SDC file” box and enter the name of the SDC file to be updated. The file should contain valid SDC commands for the current design.

Click in the “Output SDC file” box and enter the name of the resulting SDC output file into which the updated SDC timing constraints are written.

- Load back updated SDC file – Select this option to reload the updated SDC file. The updated SDC constraints are also automatically reloaded into the CEL view.
- Name Prefix – Select this option to specify the prefix to use when naming flattened cells.

Simple name – Select this option to indicate that simple, nonhierarchical names are to be used for cells that are flattened. The cells maintain their original names. This is the default.

No backslash – Select this option to indicate that a backslash should not be added before the hierarchy delimiter.

Specified prefix – Select this option to specify the prefix to be used when naming flattened cells. The naming style is:

<specified_prefix><original_cell_name>

3. Click OK or Apply.

Automatically Placing and Shaping Objects In a Design Core

You can automatically place and shape plan group boundaries, black boxes, voltage areas, and other soft macros in a design core.

This section includes the following topics:

- [Keeping the Top-level Area of the Design Contiguous During Shaping](#)
- [Using Variables or Parameters to Control Channel Resizing](#)
- [Using Relative Placement Constraints to Guide the Placement and Shaping of Objects](#)

Plan groups are automatically shaped, sized, and placed inside the core area based on the distribution of cells resulting from the initial virtual flat placement. Blocks (plan groups, voltage areas, and soft macros) marked as fixed (they have the `is_fixed` property set to `true`) remain fixed; the other blocks, whether or not they are inside the core, are subject to being moved or reshaped.

Note:

It is assumed that an initial virtual flat placement has been run on the design.

To automatically place and shape objects in the design core,

1. Choose Placement > Place and Shape Plan Groups.

The Place and Shape Plan Groups dialog box appears.

Alternatively, you can use the `shape_fp_blocks` command.

2. Select the “Place and Shape plan groups” option. The default is on.

3. Select the remaining options as needed.

- Prefer rectilinear shapes – If you do not select this option, the `shape_fp_blocks` command creates only rectangular plan groups or soft macros boundaries unless they would overlap fixed objects, in which case the shape might need to be rectilinear if it needs to be cut out around fixed objects.

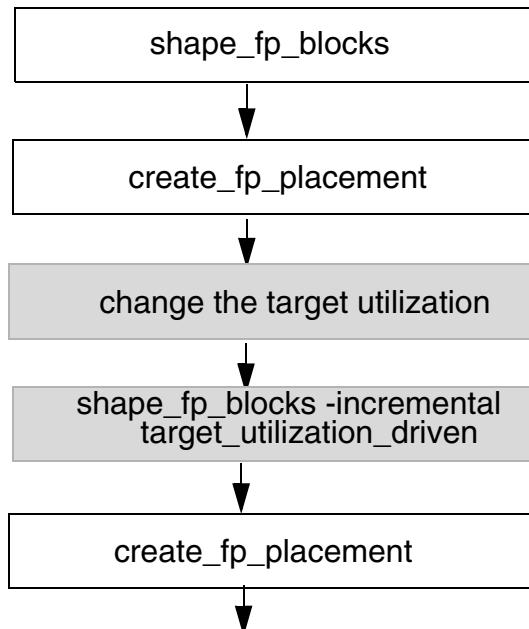
If you select this option, the virtual flat placer tries to put the plan group boundary around the set of preplaced cells associated with that plan group. However, in cases where you have one small block and one large block in your design, the small block is placed by one of the corners of the large block, creating a rectilinear L-shaped boundary. L-shaped plan groups can be created even if no fixed objects are encountered on the floorplan. The default is off.

- Run Incrementally – When you select this option, the floorplan is adjusted to meet the new plan group or voltage area target utilizations or to reduce the placement congestion in channels between blocks or inside the blocks themselves. The input is a legal, nonoverlapping floorplan and a congestion map if you select the congestion driven mode. The `shape_fp_blocks` command incrementally resizes and reshapes the blocks (plan groups, voltage areas, and soft macros) in the design, changing the floorplan as little as possible, so that the estimates based on the congestion map are satisfied.

The run incrementally option runs in two modes: target utilization driven and congestion driven.

Target utilization driven – If you select this mode, the `shape_fp_blocks` command adjusts the floorplan so that it meets the newly specified target plan group or voltage area utilizations. The command should achieve target utilizations within the `blockPlace_UtilStack` variable settings. The new floorplan should be similar to the original one. This mode is useful when you are satisfied with the overall locations of the plan groups, but realize that the sizes of the plan groups need to change.

[Figure 6-7 on page 6-16](#) shows the target utilization flow.

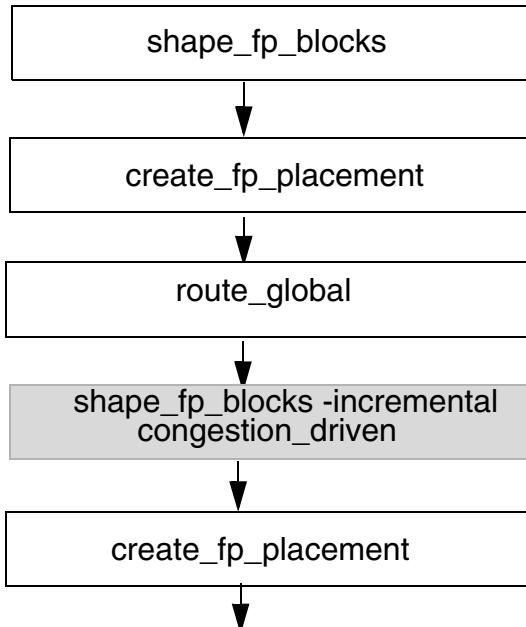
Figure 6-7 Target Utilization Shaping Flow

Congestion driven – If the channels between the blocks (plan groups, voltage areas, and soft macros) are too narrow or if the blocks themselves are too small, congestion in the design can often result. If you select this mode, the `shape_fp_blocks` command first attempts to use the congestion map created by global routing. If a congestion map is not available, it tries to use a placement congestion map. If neither of the two congestion maps is available, an error message is generated. Once a congestion map exists, you can use it to estimate the required sizes of the channels and blocks that will be needed to help reduce congestion.

The channels and blocks are resized and reshaped, changing the floorplan as little as possible, to reduce congestion in the channels between blocks and inside the blocks themselves so that the estimates based on the congestion map are satisfied. The objective is to reduce congestion by resizing channels as close to an optimal width as possible based on the existing congestion map.

[Figure 6-8 on page 6-17](#) shows the congestion-driven shaping flow.

Figure 6-8 Congestion-Driven Shaping Flow



Occasionally, standard cells move into widened channels which causes additional connections to pass through the channels and therefore, making the initial channel congestion estimates invalid. To limit the instances of this occurring, you can use the `set_app_var blockPlace_addChannelBlockages` variable to automatically control the creation of blockages in the widened channels. You can use hard placement blockages or partial blockages with the congestion density reflecting the cell area that was initially inside the channel.

Note:

Because the placement legalizer does not obey partial blockages, using hard placement blockages in the channels is generally better, but only if there is enough space.

For a description of the parameters you can use to control channel resizing, see [Table 6-1 on page 6-20](#).

- Create routing channels – If you select this option, routing channels are created between plan groups, black boxes, and soft macros. The channel widths are based on pin counts with nonperpendicular connections to objects on the associated object edge. The default is off.

If you want channels between the top-level blocks in your design, but the design is not pad limited in size, set the initial core utilization slightly lower than the initial block utilizations. A utilization of 0.65 for the core and a utilization of 0.7 for the blocks is recommended.

- Refine placement afterwards – If you select this option, the placement is refined after the placement and shaping of all unfixed plan groups, soft macros, and black boxes. (This is the equivalent of running the `create_fp_placement` command without any arguments.) The default is off.
- Top down block placement mode – If you select this option, the command sets variables before it runs so that the placement and shaping of objects is constrained like top down placement in JupiterXT (`fphBlockPlacement` command). Initial virtual flat placement is not assumed when you use this option. After the command runs, any variables that it set are restored to their previous settings. The default is off.
- Place sub macros – If you select this option, the command places hard macros in the top-level cell and in all unfixed soft macros. The default is off.
- Block small placement area – If you select this option, placement areas between blockages or fixed objects or both which are smaller than the input threshold value (in microns) are blocked out during shaping. This avoids skinny “fingers” that can occur when there are many blockages or when there are blockages with small channels between each other or the design core.

Select the “Narrower than (microns)” option and enter a number in microns. By default, the value is -1, which means that the tool automatically calculates a reasonable value for this option.

- Use placement constraint file – If you select this option, you must enter the name of the file that contains the path and name of an ASCII file containing relative placement constraints to guide the placement and shaping of objects.

The format of the constraint file is one constraint per line. The first word in the line is always the name of the constraint. For example,

`fplModuleDestRegion plan_group_name region_abbrev`

where `region_abbrev` is one of N, S, E, W, NE, NW, SE, SW

For a description of the relative placement constraints, see “[Using Relative Placement Constraints to Guide the Placement and Shaping of Objects](#)” on page 6-22.

4. Click OK or Apply.

Keeping the Top-level Area of the Design Contiguous During Shaping

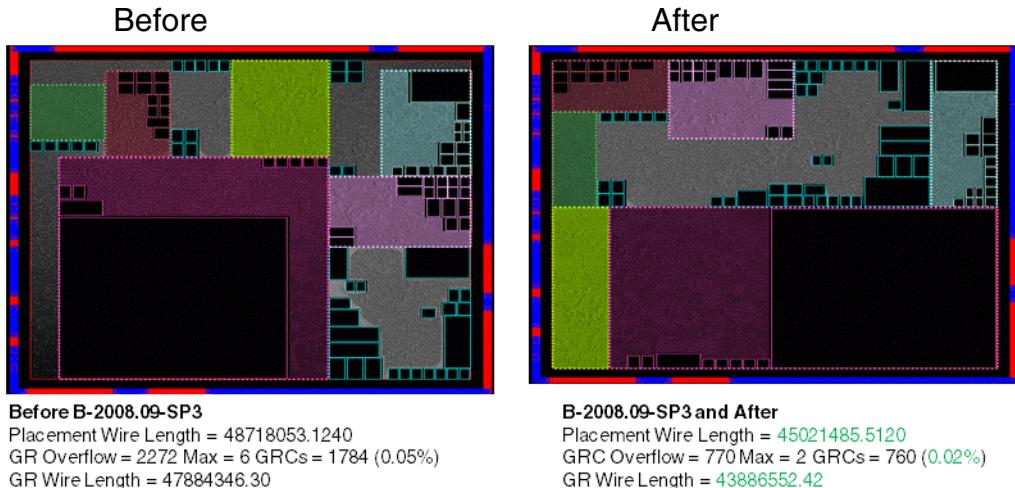
You can use the `blockPlace_keepTopLevelTogether` variable to keep the top-level area of the design contiguous during the placement and shaping of objects in the design core.

```
icc_shell > set_appr_var blockPlace_keepTopLevelTogether 0 | 1
```

Setting this variable to a value of 1, forces the `shape_fp_blocks` command to keep the top level area of the design contiguous during shaping. Keeping the top level contiguous enables the signals to reach all of the top-level area without going inside the plan groups, which is particularly useful if no feedthroughs are allowed. The default is 0.

[Figure 6-9](#) shows an example of a noncontiguous top-level area (before) and a contiguous top-level area (after) of the design.

Figure 6-9 Contiguous Top-Level Area of the Design



Using Variables or Parameters to Control Channel Resizing

Table 6-1 describes the variables and parameters used to control channel resizing.

Table 6-1 Channel Resizing Variables and Parameters

Option	Usage
<code>set parameter -module blockPlace -name HMUtil -value 1</code>	Determines how much hard macros contribute to the required area. Overall, the required area for a plan group is: $\text{standardCellArea} * (\text{targetUtil} + \text{utilSlack}) + (\text{paddedHardMacroArea} * (\text{HMUtil} + \text{utilSlack}))$ <p>where</p> $\text{targetUtil} + \text{utilSlack} \leq 1$ <p>and $\text{HMUtil} + \text{utilSlack} \leq 1$.</p> <p>Channel resizing will not shrink the plan group below its required area.</p>
<code>set blockPlace_utilSlack 0.1</code>	Controls how much channel resizing can exceed the target utilization. The default is 0.1 (10 percent). <p>Channel resizing will not exceed 100 percent utilization for plan groups. If used for black boxes, the channel resizing can exceed the target utilization.</p>
<code>set blockPlace_allowChannelShrinking 1</code>	Shrinks channels that are not congested. This enlarges the plan group.
<code>fplChannelSize plangroup1Name plangroup2Name number</code>	Specifies the exact channel size requested between two blocks. The two blocks (plan groups, soft macros, or black boxes) must be unfixed. Enter this constraint in the constraint file. Enter the name of this constraint file in the text box located on the Place and Shape Plan Group dialog box.

Table 6-1 Channel Resizing Variables and Parameters

Option	Usage
<code>set blockPlace_minChannelSize number</code>	Specifies a minimum channel size.
<code>set blockPlace_channelsRespectAspectRatio 1</code>	Specifies whether or not the command should respect the aspect ratio during channel resizing. Set it to 1 to respect aspect ratios or leave it set to 0 (the default) to allow the aspect ratios to change.
<code>set_parameter -module blockPlace -name minAspectRatio -value number</code>	Specifies the minimum aspect ratio limits.
<code>set_parameter -module blockPlace -name maxAspectRatio -value number</code>	Specifies the maximum aspect ratio limits.
<code>set blockPlace_avoidPowerGrid 1</code>	Specifies whether or not plan group boundaries should be away from the power grid. (See the <code>blockPlace_distanceToPowerGrid</code> variable for an explanation.)
<code>set blockPlace_distanceToPowerGrid number</code>	Controls the minimum allowed distance between a plan group boundary and a power grid. It has no effect if the <code>blockPlace_avoidPowerGrid</code> variable is off.
<code>set_parameter -module blockPlace -name reportLargeChannels -value 1</code>	Reports the channels that are too large.
	After the channels are resized, a report is generated. It lists the channels that are not the optimal size based on the congestion numbers, and the reasons why.
	This parameter is ignored if the <code>blockPlace_allowChannelShrinking</code> variable is set to 0.

Table 6-1 Channel Resizing Variables and Parameters

Option	Usage
<code>set_paramenter -module blockPlace -name reportFixedObjectChannels -value 1</code>	Reports the channels that have a fixed object (blockage, fixed cells, I/O pads) on one side. Channels that have fixed objects on both sides are not reported.
<code>set_appr_var blockPlace_keepTopLevelTogether 0 1</code>	When this variable is set to a value of 1, it forces the <code>shape_fp_blocks</code> command to keep the top-level area of the design contiguous (in one piece) during the placement and shaping of objects in the design core. The default is 0.
<code>set_app_var blockPlace_addChannelBlockages 0 1 2</code>	When this variable is set to a value of 0, no blockages are created to control congested channels. A value of 1 uses partial blockages to control the congested channels, and a value of 2 uses hard macro placement blockages to control congested channels. The default is 0.

Using Relative Placement Constraints to Guide the Placement and Shaping of Objects

Relative placement constraints considered in a constraint file for guiding the placement and shaping of objects are described as follows:

- `fplNamedGroup groupName`

Note:

Groups must first be defined by using the `fplNamedGroup` constraint.

If *groupName* is not a name of an existing group, this constraint creates one and places *obj* in it. *Obj* can be a name of a block or another (already created) group. The block is a soft macro, black box, or plan group. Grouping, in the context of these relative constraints, can be viewed as a “pseudo hierarchy” because these are physical groups that do not have to be related to the logic hierarchy of the netlist.

- `fplRel relType obj1 obj2`

relType (a type of relation)

Supported types are

- `BT` (bottom top)–This means *obj1* should be below *obj2*.
- `LR` (left right)–This means *obj1* is to be placed to the left of *obj2*.

Care should be taken to ensure consistency of these relative constraints. There is no constraint “conflict resolution” up front when running the `shape_fp_blocks` command. For example, do not use “`fplRel BT o1 o2`” and “`fplRel BT o2 o1`” or other, more complicated and conflicting constraint sets. Likewise, constraining some elements of group1 above elements of group2 while placing other elements of group1 below elements of group2, will give unpredictable results. Do not make relationships between an object and its group. Generally, relations should be on the same grouping level.

- `fphModuleDestRegion plan_group_name region_abbrev`
where *region_abbrev* is one of N, S, E, W, NE, NW, SE, SW

Here is an example constraint file:

```
fplNamedGroup red plan_group1
fplNamedGroup red black_box_foo

fplNamedGroup blue plan_group2
fplNamedGroup blue plan_group3
fplNamedGroup green plan_group4
fplNamedGroup green black_box_bar

fplNamedGroup root red
fplNamedGroup root blue
fplNamedGroup root green

fplRel BT plan_group1 black_box_foo
fplRel LR plan_group3 plan_group2

fplRel BT green blue
fplRel LR red green
```

Enabling Plan Group-Aware Scan Chain Reordering

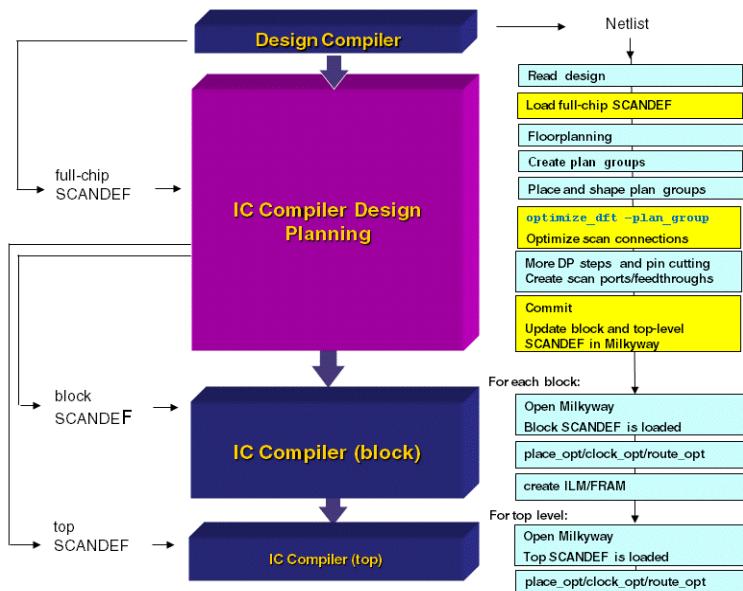
In IC Compiler design planning, you can use the `read_def` command to load the full-chip expanded SCANDEF information at the beginning of the design planning stage to prevent the scan chain elements from being reoptimized during in-place optimization. Reading the full-chip SCANDEF information enables scan chain elements to have `dont_touch` or `size_only` attributes applied. As a result, the scan chain order is maintained through the design planning stage.

DFT Compiler uses the `write_scan_def -expand list_of_instances` command at the chip-level to write the SCANDEF file with all the scan cells of the subblocks.

In the DFT-aware design planning flow, you can use the `optimize_dft -plan_group` command to optimize scan chains based on the physical hierarchy after you place and shape the plan groups and before you perform pin cutting. This is useful when logical hierarchy-based scan chains are suboptimal for the best physical design. The plan group-aware scan chain reorders the scan cells to reduce the number of scan chain ports and feedthroughs to reflect the physical hierarchy and virtual flat placement results.

[Figure 6-10](#) shows where the `optimize_dft -plan_group` command is used in the DFT-aware design planning flow.

Figure 6-10 DFT-Aware Design Planning Flow



During commit hierarchy (`commit_fp_plan_groups` command), the SCANDEF information is updated, based on the plan groups. The original logical hierarchy-based SCANDEF information is automatically updated to correspond to the physical hierarchy in the design planning phase.

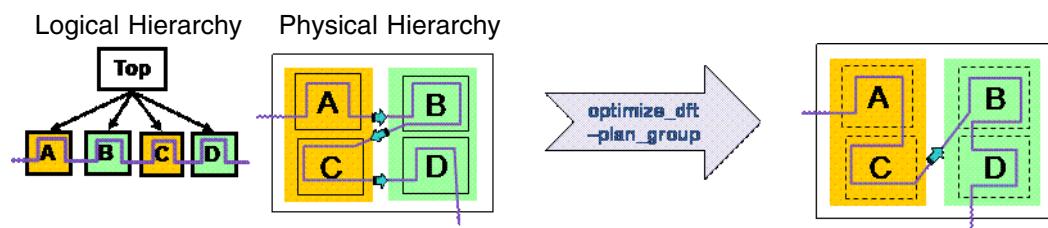
Uncommitting the physical hierarchy (`uncommit_fp_soft_macros` command) is not supported.

Using Plan Group-Aware Scan Reordering

Using the `optimize_dft -plan_group` command provides the following objectives and benefits:

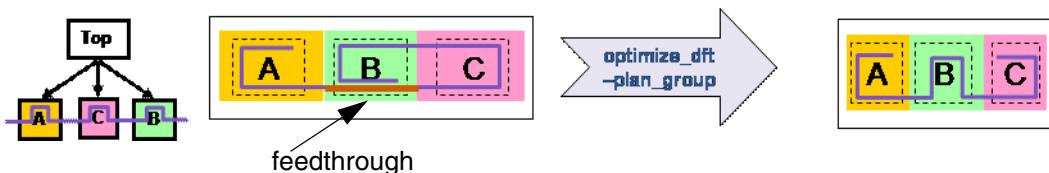
- The design is optimized to minimize the number of unnecessary top-level scan-chain wires crossing between the physical hierarchies during design planning.
- The top-level and block-level SCANDEF files are output automatically according to the physical hierarchy.
- Congestion is minimized and routability is improved.
- During scan chain reordering, the scan cells of the same plan group are clustered together, and therefore, the entry and exit points of the plan group and the top-level scan wires and ports are minimized, as shown in [Figure 6-11](#).

Figure 6-11 Minimizing Top-Level Scan Wires



- An optimal order of the physical hierarchies is determined for each top-level scan chain to minimize excessive feedthroughs created for scan wires, as shown in [Figure 6-12 on page 6-25](#).

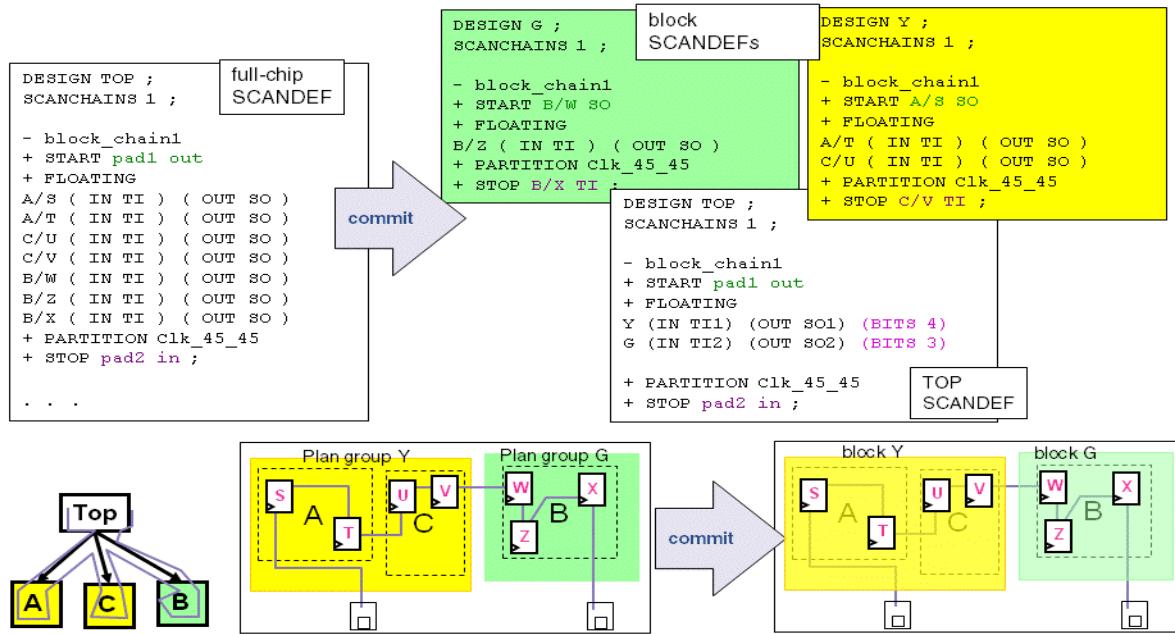
Figure 6-12 Minimizing Feedthroughs



When the physical hierarchy is committed using the `commit_fp_plan_group` command, the corresponding plan group-aware SCANDEF data is updated simultaneously, based on the optimizations performed during the plan group-aware scan chain reordering process.

During commit hierarchy, block-level SCANDEF information is pushed down into the hierarchy, and the updated SCANDEF is stored in the Milkyway CEL views for the blocks and top level, based on the physical hierarchy, as shown in [Figure 6-13](#).

Figure 6-13 Updating SCANDEF Data During Commit Hierarchy



Checking the Consistency of the SCANDEF Data Against the Netlist

IC Compiler maintains the SCANDEF data that describes the scan-chain characteristics and constraints, as well as the scan-stitched netlist. The netlist and SCANDEF data must be consistent with each other. To validate their consistency, you can use the `check_scan_chain` command. By default, a summary of the consistency of all the chains is displayed.

Consistency checking is performed on elements between the START and STOP points of the chain stubs. The check starts at the STOP point and proceeds backward. Each scan chain stub is given a status after the checks are complete.

Using Plan Group-Aware Repartitioning

You can use the following command to perform plan group-aware repartitioning of the scan chains during design planning.

```
set_optimize_dft_options -repartitioning_method none | single_directional
| multi_directional | adaptive
```

For more information about this command and options, see the man page.

Note:

You can turn off repartitioning and perform only plan group-aware scan chain reordering by using the following setting prior to running the `optimize_dft -plan group` command.

```
set_optimize_dft_options -repartitioning_method none
```

Plan group-aware repartitioning provides the following objectives and benefits:

- It can further reduce the scan chain wires crossing between plan groups above what plan group-aware scan chain reordering alone can do.
- It reduces implementation complexity by reducing feedthrough nets along the scan paths.
- It reduces the top-level routing congestion.

Adding Padding to Plan Groups

To prevent congestion or DRC violations, you can add padding around plan group boundaries. Plan group padding sets placement blockages on the internal and external edges of the plan group boundary to prevent cells from being placed in the space around the plan group boundaries. Internal padding is equivalent to boundary spacing in the core area. External padding is equivalent to macro padding.

You can pad the plan groups to ensure that the placed standard cells do not extend over the plan group boundary and that cells belonging to the plan group remain inside the plan group, while the cells that do not belong to the plan group remain outside the plan group when the physical hierarchy is committed. The boundary space is also needed for pins and the routing to those pins.

Note:

You should pad plan groups before refining the placement to ensure that the placement honors the plan group boundaries.

Padding the plan groups also reserves space for the pins on soft macros, which in turn, are created from the plan groups.

When the physical hierarchy is committed, the padding is transferred to a soft macro cell as core-boundary spacing and not as placement blockages in the child soft macro.

The plan group padding is visible in the layout window. The plan group padding is dynamic, which means that it follows the changes of the plan group boundaries.

To add padding to plan groups,

1. Choose Floorplan > Create Plan Group Padding.

The Create Plan Group Padding dialog box appears.

Alternatively, you can use the `create_fp_plan_group_padding` command.

Alternatively, you can use the `create_fp_plan_group_padding` command.

2. Specify the plan groups for which you want to add padding.

- All – Select this option to add padding for all plan groups. This is the default.
- Specified – Select this option to add padding for selected plan groups.

Enter the names of the plan groups.

3. Specify whether the padding is internal, external, or both.

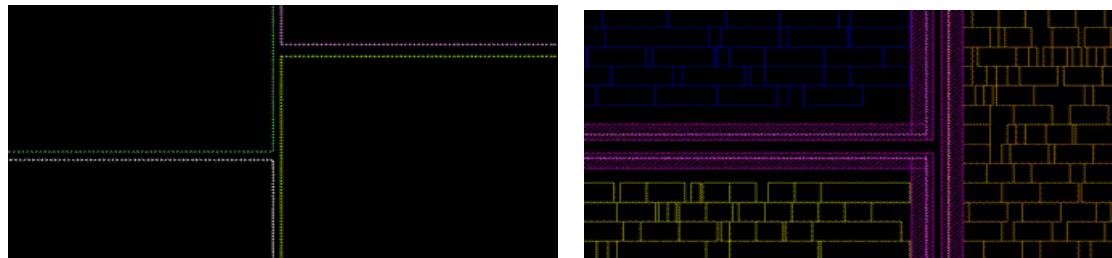
- Internal – Select this option to add padding inside the plan group boundaries. Enter a width value. Internal padding should be at least 1 micron (the default) wide to allow space for pins and pin routing.
- External – Select this option to add padding outside the plan group boundaries. Enter a width value. The default is 0.

If you want to add both internal and external padding, select the “Internal” and “External” options and enter the width values.

4. Click OK or Apply.

[Figure 6-14](#) shows an example of with and without plan group padding.

Figure 6-14 With/Without Plan Group Padding



Removing the Plan Group Padding

To remove (delete) both external and internal padding for the plan groups, choose Floorplan > Delete Floorplan Objects > Delete Plan Group Padding. The Delete Plan Group Padding dialog box appears.

Alternatively, you can use the `remove_fp_plan_group_padding` command.

You can delete padding for selected plan groups or for all plan groups. The default is all plan groups.

Adding Block Shielding to Plan Groups or Soft Macros

When two signals are routed parallel to each other, crosstalk can occur between the signals, leading to an unreliable design. You can protect signal integrity by adding modular block shielding to plan groups and soft macros.

The shielding creates rectangular metal layers around the outside of the soft macro boundary or plan group in the top level of the design, and around the inside boundary of the soft macro or plan group, or both.

Usually, you can selectively create metal layers to block the router from routing long nets along the block boundaries. Note however, that sometimes, even if you block layers in the preferred direction, nets can still be routed along the block boundaries. In that case, you can consider blocking all layers. The block shielding is created along the soft macro boundaries, but it leaves narrow openings to access the pins.

To handle signal integrity when creating hierarchical designs, the router must be prevented from laying down routes that run collinear to the soft macro or plan group boundary. To do this, rectangles that the router creates are placed along and inside the soft macro's or plan group's edges on metal layers whose preferred direction crosses the soft macro or plan group boundary. If the plan groups are committed, the inner-boundary shielding is then transferred to the soft macro.

The rectangles act as hierarchical signal shielding by allowing only routes that are perpendicular to the soft macro or plan group boundary to pass through the metal layers. By preventing collinear routing along the soft macro and plan group boundaries, signal crosstalk among routing in the soft macro and routing in the top level channels is minimized.

To add block shielding for plan groups or soft macros,

1. Choose Floorplan > Create Module Block Shielding.

The Create Module Block Shielding dialog box appears.

Alternatively, you can use the `create_fp_block_shielding` command.

2. Select the type of objects to which you want to add signal shielding.

You can add shielding for all plan groups and soft macros, or you can specify selected plan groups or soft macros. The default is “All”.

3. Select a scope option.

- Inside – Select this option if you want to create metal route blockages (shielding) around the boundary inside the plan groups or soft macros.
- Outside – Select this option if you want to create metal route blockages (shielding) around the boundary outside the plan groups or soft macros.
- Both – Select this option if you want to create metal route blockages (shielding) around the boundary both inside and outside the plan groups or soft macros. This is the default.

4. Specify a shielding width.

Click in the Width box and enter a value. The shielding width is determined by multiplying the layer pitch with the value you specify. The default multiplier is 3.0. You can also specify a value in microns.

5. Select the metal layers.

Specify the metal layer or layers on which the rectangles (shielding) will be created. If no layers are specified, the command creates shielding on each metal layer and uses the layer's preferred direction to determine on which sides shielding is created.

For example, if metal 2's preferred direction is horizontal, then metal 2's rectangle blockages are created on the top and bottom sides of the plan group and/or soft macro.

6. Specify the sides on which to create the shielding.

Select the sides (Left, Right, Top, or Bottom) on which to create the shielding rectangles. If you do not specify a side, shielding is created on all sides of the plan groups or soft macros.

7. Click OK or Apply.

Removing Module Block Shielding

You can remove the signal shielding (route blockages) created by modular block shielding. Choose Floorplan > Delete Floorplan Objects > Delete Module Block Shielding. The Delete Module Block Shielding dialog box appears.

Alternatively, you can use the `remove_fp_block_shielding` command.

You can remove shielding rectangles from all plan groups and soft macros or from specified plan groups or soft macros. You can also select the sides as well as the metal layers on which shielding should be removed.

7

Performing Power Planning

After you have completed the design planning process and have a complete floorplan, you can perform power planning.

This chapter includes the following sections:

- [Editing Power and Ground and Tie-Off Connections for Nonmultivoltage Designs](#)
- [Adding Power and Ground Rings](#)
- [Adding Power and Ground Straps](#)
- [Prerouting Standard Cells](#)
- [Setting Preroute Design Rule Checking Options](#)
- [Defining a Set of Special Preroute Rules](#)
- [Setting Preroute Advanced Via Rules](#)
- [Creating Preroute Vias](#)
- [Performing Low-Power Planning for Multithreshold-CMOS Designs](#)
- [Performing Power Network Synthesis On Multivoltage Designs](#)
- [Performing Power Network Synthesis](#)
- [Analyzing the Power Network](#)
- [Viewing the Analysis Results](#)

- [Summary of Power Network Analysis and Power Network Synthesis Options](#)
- [Running PrimeRail Within IC Compiler](#)

Editing Power and Ground and Tie-Off Connections for Nonmultivoltage Designs

You can edit power and ground (PG) and tie-off connections for nonmultivoltage designs.

A tie-off connection is the connection between a signal pin and a PG net. If a signal pin is connected to 1'b0 or 1'b1 or to a PG net, it is direct-tie-off connection. If a signal pin is connected to a net that is assigned to 1'b0 or 1'b1 or to a PG net, it is an indirect-tie-off connection.

When IC Compiler reads in a Verilog netlist for a direct tie-off connection, it maps the 1'b0 or 1'b1 to Milkyway database default SNPS_LOGIC0 or SNPS_LOGIC1 nets in the same hierarchy. When IC Compiler reads in a Verilog netlist for an indirect tie-off connection, the net is derived as `net_type` of tie-high or tie-low.

In the following verilog input example, pin TOP/inv_inst1 is an indirect tie-off pin because it is connected to net1, which is a net that is assigned to 1'b0, and pin TOP/inv_inst2 is a direct tie-off pin because it is connected directly to 1'b0.

```
Module TOP ( in , out );
  inout in;
  output out;
  wire net1;
  assign net1 = 1'b0;
  INV inv_inst1 ( .I ( net1 ), .Z ( ) );
  INV inv_inst2 ( .I ( 1'b0 ), .Z ( ) );
endmodule
```

Table 7-1 summarizes the PG tie-off editing commands that are used in nonmultivoltage designs.

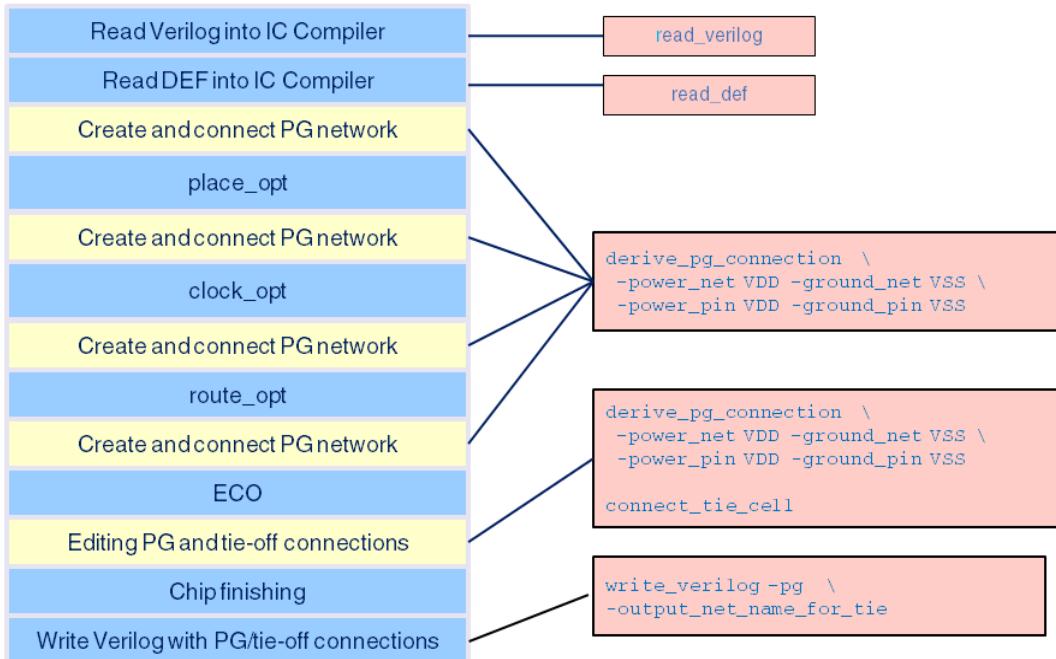
Table 7-1 Summary of PG Tie-Off Editing Commands

Command	Description
derive_pg_connection	<p>Performs the automatic connection of power and ground pins and direct rail-tie connections. For a single voltage design (nonmultivoltage), user-provided power and ground nets are used for automatic connections.</p> <p>For a multivoltage design, the connections are derived from the power domains defined in the multivoltage design.</p> <p>By default, this command works only on the unconnected power and ground nets.</p>
connect_net	Connects the specified net to the specified pins or ports for PG and tie-off editing and for signal editing at the same hierarchical level.
disconnect_net	Breaks the connections between a net or a net instance and its pins or ports. The net, pins, and ports are not removed.
connect_tie_cells	Specifies the cell ports from which to connect tie-high and tie-low cells to the specified tie-off inputs of other cells at the same hierarchical level.
recover_tie_connection	Recovers tie-off connections from PG nets to SNPS_LOGIC0 or SNPS_LOGIC1 nets for tie-off optimization, recovers the tie-off connections for a new tie-off implementation, or preserves the tie-off connections before removing the PG nets or the SNPS_LOGIC0 or SNPS_LOGIC1 nets.
report_tie_nets	Reports the number of tie-high nets and tie-low nets and the connected pin and port for each tie net in the current design.
report_pg_net	Reports information about the power and ground nets for the currently opened Milkyway design.
create_net	Creates power and ground nets, tie-off nets, as well as signal nets in the current design

Table 7-1 Summary of PG Tie-Off Editing Commands

Command	Description
create_pg_network	Creates a new hierarchical power or ground network at any level of hierarchy. The newly created PG network connects multiple hierarchical cell instances across selected hierarchical boundaries.
remove_pg_network	Removes hierarchical power and ground connections from specified hierarchical cell instances and their children or from the entire design.
remove_tie_cells	Removes the tie cells specified by a given collection. The signal pins that were originally driven by the deleted tie cells are connected to the default Milkyway tie-high or tie-low nets.

Figure 7-1 shows the recommended PG and tie-off editing for nonmultivoltage designs with tie cells in the library.

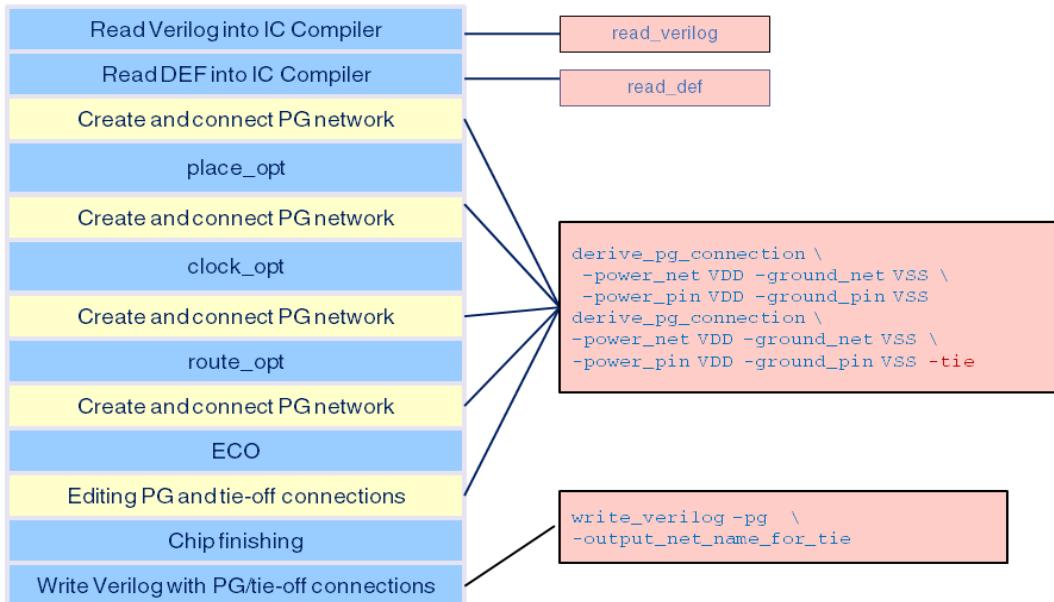
Figure 7-1 PG and Tie-Off Editing Design Flow With Tie Cells in the Library

For designs that use tie cells, meaning that the tie-off pins are connected to power and ground nets through the tie cells, the IC Compiler optimization `place_opt` and `psynopt` commands, automatically insert tie cells for tie-off connections. However, because no

optimization commands are run after an ECO, you must use the `connect_tie_cells` command to manually insert tie cells for any tie-off connections that are changed during an ECO.

[Figure 7-2](#) shows the recommended PG and tie-off editing for nonmultivoltage designs without tie cells in the library.

Figure 7-2 PG and Tie-Off Editing Design Flow Without Tie Cells in the Library



For designs that do not use tie-cells, meaning that the tie-off pins are connected directly to the power and ground nets, you should update the PG/tie-off connection after each major command is run, as shown in the design flow in [Figure 7-2](#). This is recommended to ensure that after you run each major command, the design is clean and has the correct PG/tie-off connection.

This section includes the following topics:

- [Performing Automatic Connections of Hierarchical Power and Ground Pins](#)
- [Creating a New Hierarchical PG Network or Updating an Existing PG Network](#)
- [Removing Hierarchical Power and Ground Connections](#)
- [Manually Connecting Power and Ground and Tie Nets to Pins or Ports](#)
- [Recovering a Tie-Off Connection From a Direct PG Connection](#)
- [Recovering a Tie-Off Connection From a Direct PG Connection](#)
- [Removing Tie Cells](#)

- [Reporting Power and Ground Nets](#)

Performing Automatic Connections of Hierarchical Power and Ground Pins

You can perform automatic power and ground connections for the power and ground pins of cells in the design, as well as direct rail-tie connections to power and ground nets by using the `derive_pg_connection` command. This is the only recommended method for creating a power and ground (PG) network for nonmultivoltage designs. Use this command before using any of the optimization commands.

The design can be a single voltage design (nonmultivoltage) with one set of user-defined power and ground nets used for automatic connections or a multivoltage design with connections derived from the power domains defined in the multivoltage design.

In the case of a single voltage design, the `-power_net name` and `-ground_net name` options are required.

In the case of a multivoltage design, each power domain should have at least a permanent power net and permanent ground connections.

The automatic connection behavior for power pins is as follows:

- For a power pin where an exception connection is defined, the power net specified in the exception is used.
- For cells with a single power pin that are used as always-on cells, the backup power net is used.
- For cells with a single power pin that are *not* used as always-on cells, the primary power net is used.
- For multithreshold-CMOS switch cells with multiple power pins, the power pins are connected to the power domain power nets of the same type – that is, primary power pin to primary power net and internal power pin to internal power net.

Note:

The rules applicable for the automatic connection of ground pins are similar to those for the automatic connection of power pins.

The `derive_pg_connection` command uses the following syntax:

```
derive_pg_connection [-reconnect] [-tie] [power_net name] [-ground_net name] [-power_pin name] [-ground_pin name] [-cells cell_list] [-create_port top | all | none] [-all] [-create_nets] [-verbose]
```

Use this option...	To do this...
<code>-reconnect</code>	Forces the command to work on the entire power network by reconnecting the already-connected power and ground pins as well as connecting the unconnected pins.
<code>-tie</code>	<p>Perform direct rail-tie-off connections to power and ground nets in the design.</p> <p>For a multivoltage design with power domains, the tie pins are connected to the proper power nets consistent with their cell instances' power connections.</p> <p>For a design without a power domain, you must explicitly specify a <code>-power_net name</code> and <code>-ground_net name</code> as the tie-off nets.</p>
<code>-power_net name</code>	Specify the name of the power net to use for power and tie-high connections for designs without power domain information.
<code>-ground_net name</code>	Specify the name of the ground net to use for ground and tie-low connections for designs without power domain information.
<code>-cells cell_list</code>	<p>Specify a list of cells to connect to power and ground nets.</p> <p>Use this option only when a subset of the cell instances is needed.</p>
<code>-power_pin name</code>	Specify the name of the power pin on cells specified by the <code>-cells</code> option.
	Use this option when there are multiple power pins on a cell and each pin needs to connect to a different power net.
<code>-ground_pin name</code>	<p>Specify the name of the ground pin on cells specified by the <code>-cells</code> option.</p> <p>Use this option when there are multiple ground pins on a cell and each pin needs to connect to a different ground net.</p>

Use this option...	To do this...
<code>-create_port top all none</code>	Create power and ground I/O ports if they do not already exist for the nets specified with the <code>-power_net</code> or <code>-ground_net</code> options. The <code>-create_port top</code> option creates top-level power and ground ports only. The <code>-create_port all</code> option traverses the physical hierarchy and creates ports on child cells (soft macros and black boxes). The default is <code>none</code> (no new ports are created).
<code>-all</code>	Perform automatic connections of power and ground pins, and tie-high and tie-low rails simultaneously.
<code>-create_nets</code>	Create power and ground nets based on power domain definitions and supply nets.
<code>verbose</code>	Report the details of the changes to the power and ground connections.

Using the GUI Menu Command to Perform Power and Ground Connections

You can also use the GUI menu command to perform automatic power and ground connections for the power and ground pins of cells in the design as well as direct rail-tie connections to power and ground nets.

To perform automatic power and ground and tie connections,

1. Choose Preroute > Derive PG Connection.

The Derive Power Ground Connection dialog box appears.

2. Set the options, depending on your requirements.

- Auto connection – Performs automatic power and ground connections for the power and ground pins of cells in the design as well as direct rail-tie connections to the power and ground nets. This is the default.
- Reconnect power/ground pins with existing connections – Uses existing connections to reconnect all power and ground pins in the design.
- Create power/ground nets from UPF supply nets – Creates the power and ground nets based on power domains defined in the multivoltage design and from UPF supply nets.

- Perform both power and tie connections – Performs direct rail tie-off connections in the design. For multivoltage designs with power domains, the tie pins are connected to the proper power nets consistent with their cell instances' power connections. For a single voltage design, the power and ground nets specified in the “Power net” and “Ground net” text boxes are used as tie-off nets.
- Show detailed connection information – Displays detailed information of the changes to the power and ground connections.
- Manual connections – Performs manual power and ground connections for designs with no power domain information.

Enter the names of the power and ground nets to use for the power and tie-high connections and the ground and tie-low connections.

Enter the names of the power and ground pins on cells to connect to the specified power and ground nets.

- Create port – Creates I/O ports for the specified power and ground nets. Select “Top” to create top-level power and ground ports only. Select “All” to create ports on the child cells (soft macros and black boxes). The default is “None” (no ports are created).
- Cells – Enter a list of cells to connect to the specified power and ground nets.

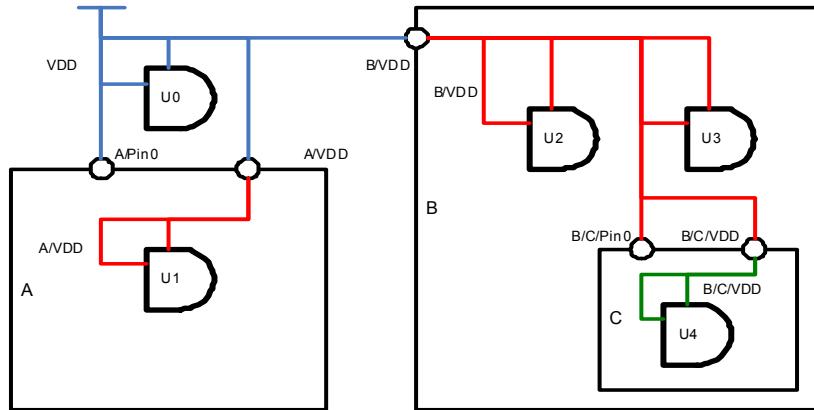
3. Click OK or Apply.

Example: Connecting Power and Ground Pins to Power and Ground Nets

After reading in the Verilog netlist by using the `read_verilog` command, and if necessary the DEF file, use the following command to automatically connect the power and ground pins to power and ground nets.

```
derive_pg_connection -power_net name -ground_net name
```

[Figure 7-3 on page 7-10](#) shows the same circuit as [Figure 7-5 on page 7-11](#), but now connected to power. Note that the hierarchical ports A/Pin0 and B/C/Pin0 are not reused. However, the hierarchical ports A/VDD, B/VDD, and B/C/VDD can be reused.

Figure 7-3 Circuit Connected to Power

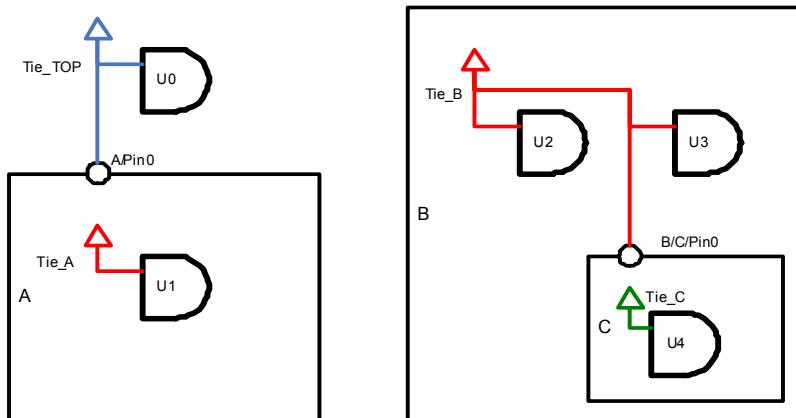
Example: Connecting Tied-Off Pins to Power and Ground Pins

The `derive_pg_connection` command will not touch any power and ground tie-off pins that are already connected to a PG network in the design; it only connects unconnected PG pins and maps tie-off nets to the real PG network, if you specify the `-tie` option.

Note:

Make sure the PG network already exists in the design before you run the `derive_pg_connection -tie` command. If you run the `derive_pg_connection -tie` command before the PG network is created, it can cause unexpected results for the PG tie-off connections in the design flow.

[Figure 7-4](#) shows an example of signal pins connected to tie nets. The cells U0 through U4 have pin called Pin0 that is tied-high to power. The hierarchical ports A/Pin0 and B/C/Pin0 are also tied-high. Each port is connected to a tie net in its hierarchy.

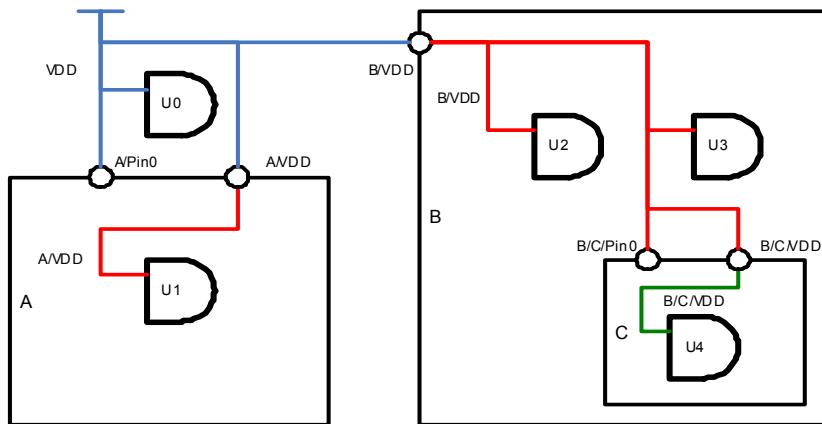
Figure 7-4 Signal Nets Connected to Tie Nets

[Figure 7-5 on page 7-11](#) shows the tied-high pins connected to VDD after the following command is issued:

```
derive_pg_connection -tie
```

The previously existing hierarchical ports A/Pin0 and B/C/Pin0 will not be reused. Instead, new hierarchical ports A/VDD, B/VDD, and B/C/VDD are created. Hierarchical nets VDD, A/VDD, and so forth are also created. The tie nets are deleted from the Milkyway database.

Figure 7-5 Signal Nets Connected to the Power Net



Creating a New Hierarchical PG Network or Updating an Existing PG Network

You can use the `create_pg_network` command to create a new hierarchical power or ground network at any level of hierarchy. You can also use the `create_pg_network` command to update the existing power or ground network by adding new hierarchical cell instances. The newly created power or ground network connects multiple hierarchical cell instances across selected hierarchical boundaries.

The `create_pg_network` command does not connect leaf pins. You must use the `connect_net` command to manually connect the leaf power and ground pins (see “[Manually Connecting Power and Ground and Tie Nets to Pins or Ports](#)” on page 7-14).

The `create_pg_network` command does not create top-level ports. If a port or net of the same name already exists in the specified hierarchical cell, an error message is issued.

Note:

To import power and ground connectivity from the Verilog netlist, the power or ground network must be connected at the top level. Otherwise, the `derive_pg_connection` command will not recognize or reuse the existing power and ground network created by the `create_pg_network` command or the power and ground nets created by the `create_net` command.

The `create_pg_network` command uses the following syntax:

```
create_pg_network
[-net net]
[-create_net net_name [-power] [-ground]]
cell_list
```

- If you want to create a new net for an existing power and ground network, use the `-net net` option. The net must be a power or ground net or a single-net. It can reside at any level in the hierarchy and must exist in the current design. The net type is derived from the other nets in the same power and ground network. Existing power and ground nets can include nets created by the `create_net` and `read_verilog` commands.

The `-net` argument is optional. If you do not specify it, you must specify the `-create_net` option.

- If you want to create a new net for a new power and ground network, use the `-create_net net_name` option. You must specify the name of the Milkyway net that will be created to connect the specified hierarchical cell instances.

If you use this option, you must also specify either a power net type (`-power` option) or a ground net type (`-ground` option). However, both options cannot be used simultaneously.

Note:

The `-create_net` argument is optional. If you do not specify it, you must specify the `-net` option.

- To specify a list of hierarchical cell instances to which the power and ground network is created, use the `cell_list` option.

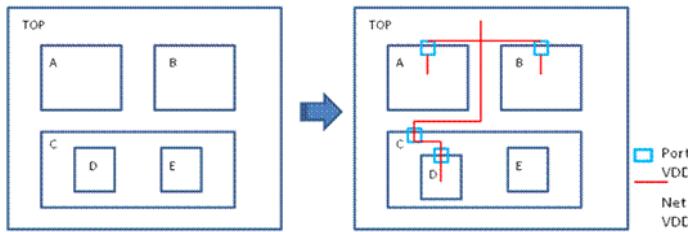
For each hierarchical cell instance that is specified, a new port is created and set to the inout port direction. The name of the port is specified by the `-create_net` option or the name of the port is the same as the net specified by the `-net` option.

The following command creates a new power and ground network across hierarchical boundaries.

```
create_pg_network -create_net VDD -power {A B C/D}
```

[Figure 7-6 on page 7-13](#) shows how the power and ground network is created in this example.

Figure 7-6 Creating a New Power and Ground Network



As shown in [Figure 7-6](#), the following occurs:

- Ports VDD are created in hierarchies A, B, and C/D.
- The hierarchical nets VDD are created and connect the new ports together across hierarchies.
 - The common parent cell is TOP. This is the hierarchical cell that is the parent cell to all specified hierarchical cells.
 - The new nets created are VDD, A/VDD, B/VDD, C/VDD, and C/D/VDD.
- The collection of hierarchical cells is {A B C C/D}.

Removing Hierarchical Power and Ground Connections

You can use the `remove_pg_network` command to remove hierarchical power and ground connections from specified hierarchical cell instances and their children. If you do not specify the hierarchical cell instances, the hierarchical power and ground connections are removed from the entire design.

The `remove_pg_network` command uses the following syntax:

```
remove_pg_network
-net list_of_pg_net
hier_cell_list | -top
```

- To specify an existing hierarchical power or ground net for removal from the power and ground network, select the `-net list_of_pg_net` option and specify the full name of the net you want to remove. You must specify only one net for removal, and it must be a single-bit net. The net can be at any hierarchical level as long as it belongs to the intended power and ground network.

- To specify a list of hierarchical cell instances from which to remove the power and ground connections, select the *hier_cell_list* option. The IC Compiler tool will remove the power and ground network from the specified hierarchical cells and their children. If the cells you specify do not exist in the current design or they are not hierarchical cells, the tool issues an error message and quits.

For each specified hierarchical cell and its children, the `remove_pg_network` removes the hierarchical power and ground connections by performing the following tasks:

- Disconnects the power and ground nets from the leaf pins that are inside the hierarchical cell instances.
- Deletes the power and ground nets that connect to the pins that are inside the specified hierarchical cell instances.
- Disconnects the power and ground nets from the ports of the specified hierarchical cell instances.
- Deletes the ports of the specified hierarchical cell instances that are connected to the power and ground network.
- Although the `remove_pg_network` command supports the removal of the PG network under specified hierarchy cells, removing a partially-built PG network is not recommended. To remove the complete PG network in the design, select the `-top` option, and rebuild a new one by using the `derive_pg_connection` command.

Note:

The `remove_pg_network` command does not remove the top-level port and its connected net. To remove the top-level power and ground port, use the `remove_port` command.

For designs without tie cells, the `remove_pg_network` command recovers the direct rail tie-off connections from the specified PG network and removes them “under the hood”; therefore, it is not necessary to run the `recover_tie_connection` command.

Manually Connecting Power and Ground and Tie Nets to Pins or Ports

You can use the `connect_net` command to connect specified pins or ports to power and ground or tie nets at the same hierarchical level. The nets can be at any level of hierarchy, but the collection of pins or ports must be at the same level of logical hierarchy as the hierarchical net.

Power and ground pins can be connected only to power and ground nets. Signal pins can be connected to signal nets, tie-nets and to power or ground nets.

The `connect_net` command uses the following syntax:

```
connect_net net object_list
```

The `net` argument specifies the net to which the pins or ports are connected.

The `object_list` argument specifies a list of pins or ports. They must reside at the same hierarchical level as the specified net and exist in the current design. Each pin must be a be unconnected.

Example: Using the connect_net command for Power and Ground Nets

The following examples are shown in [Figure 7-7](#).

- Hierarchical net VDD is connected to Pin0 on cell U0 in the top hierarchy and hierarchical cell A. The hierarchical net VDD and hierarchical pin A/Pin0 must already exist. The command performs a connection of the hierarchical pin to the outside of the module in its parent level.

```
connect_net VDD {U0/Pin0 A/Pin0}
```

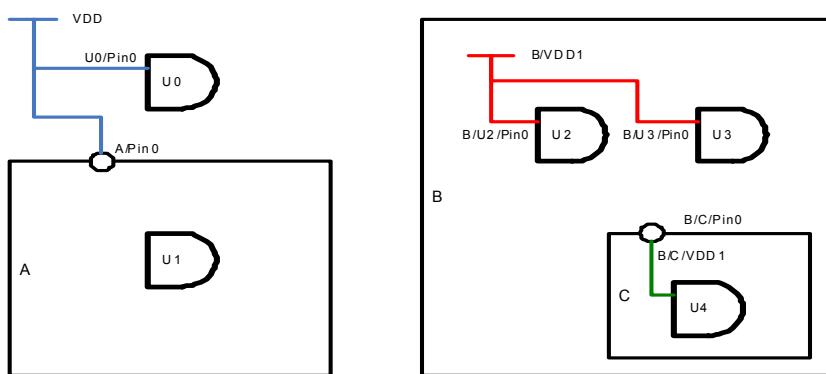
- Hierarchical net B/VDD1 is connected to Pin0 on cells U2 and U3 in hierarchy B. The hierarchical net B/VDD1 must already exist.

```
connect_net B/VDD1 {B/U2/Pin0 B/U3/Pin0}
```

- Hierarchical net B/C/VDD1 is connected to Pin0 of cell U4 in hierarchy B/C. The hierarchical net B/C/VDD1 and hierarchical pin B/C/Pin0 must already exist. The command performs a connection of the hierarchical pin inside the hierarchical module.

```
connect_net B/C/VDD1 {B/C/Pin0 U4/Pin0}
```

Figure 7-7 Examples of Power and Ground Net Connection



Performing Hierarchical Power and Ground Connections for Tie Cells

You can use the `connect_tie_cells` command to insert tie-low and tie-high cells and connect them to the specified tie-off inputs of other cells. These tie-off connections are made only to other cells at the same level of hierarchy. The power and ground pins of the tie cells are connected to the same power and ground nets used by the load cells.

A tie-low cell is a special standard cell whose output pin is always at logic low and a tie-high cell is a special standard cell whose output pin is always at logic high. A tie-high low cell is a combination cell with two output pins: one at logic high and the other at logic low.

The `connect_tie_cells` command performs the following functions:

- Automatically instantiates the minimum number of tie-high and tie-low cells needed
- Creates the necessary logical connectivity, while honoring the maximum fanout and maximum wire length constraints
- Automatically places the new cells, while honoring all placement constraints

The `connect_tie_cells` command uses the following syntax:

```
connect_tie_cells
-objects object_col1
-obj_type port_inst | cell_inst | lib_cell
[-tie_high_lib_cell lib_cell]
[-tie_low_lib_cell lib_cell]
[-tie_highlow_lib_cell lib_cell]
[-tie_low_port_name port]
[-max_fanout number]
[-max_wirelength number]
[-incremental true | false]
```

Note:

Before you run this command, your design must be fully placed and connected to power and ground.

Recovering a Tie-Off Connection From a Direct PG Connection

You can disconnect a direct PG network connection to a tie pin or port by using the `recover_tie_connection` command and then reconnect that tie-pin or port to a tie-net. You can use this command to recover the tie-off connections from the power and ground nets or tie-off optimization, to recover the tie-off connections for a new tie-off implementation or to preserve the tie-off connections before removing the power and ground network.

The `recover_tie_connection` command uses the following syntax:

```
recover_tie_connection [-net pg_net_list] [-cell hier_cell_list]  
[-hierarchy_based]
```

Use the `-net` option to recover the tie-off connection of a specified power and ground net. You can specify a list of power or ground nets. The default recovers the tie-off connection for all power and ground nets. The power and ground net can be a flat net or a hierarchical net.

Use the `-cell` option to recover the tie-off connection within a specified hierarchy cell. You can specify a list of hierarchical cells. The default recovers the tie-off connection for all the hierarchical cells and their children in the design.

Use the `-hierarchy_based` option to recover the PG tie-off connections to all tie pins or ports within the specified hierarchical cells. This option must be specified with the `-cell` option.

Reporting Tie-Off Connections

You can use the `report_tie_nets` command to determine the tie-off connections in your design. The command reports the number of tie-high nets and tie-low nets, and the connected pin and port for each net.

Removing Tie Cells

You can remove the tie-off cells from the design by using the `remove_tie_cells` command. You specify a list of tie cell instances, and the command removes them. The signal pins formerly driven by the tie cells are reconnected to the Milkyway tie-low and tie-high nets. By default, the new tie nets are the same as the original nets driven by the tie cells. However, if you use the `-use_default_tie_net` option, the original nets driven by the tie cells are deleted, and the tie nets are the Milkyway database default SNPS_LOGIC0 and SNPS_LOGIC1 nets.

The `remove_tie_cells` command uses the following syntax:

```
remove_tie_cells [-use_default_tie_net] tie_cell_list
```

Note:

If a specified cell does not exist in the design, or it is not a valid tie cell (a tie cell must be a leaf cell instance), the tool issues a warning message for the cell, and then continues to process the next cell in the specified collection.

Reporting Power and Ground Nets

You can use the `report_pg_net` to obtain a report of all power and ground nets in all hierarchical levels based on the currently opened Milkyway CEL view. By default, information is reported on all the hierarchical power and ground nets in the design.

The `report_pg_net` command uses the following syntax:

```
report_pg_net [-net net_list] [-connections]
```

For each power and ground net, the command reports the following details:

- Complete net name
- Net type (power or ground)
- UPF power domain name of the net, if available
- Total number of leaf pins, the number of ports, and the number of hierarchical pins connected to the net
- Net connections, including a detailed list of the full hierarchical names of the pins and ports connected to the power and ground nets, if requested with the `-connections` option. By default, net connections are not reported.

Reporting Cell Power and Ground Pin Connections

You can report on the connections to the power and ground pins (PG) pins, as well as the connections to the signal pins by using the `report_cell` command. This provides you with an easy way to view the PG nets and PG connections in your design.

The `report_cell` command uses the following syntax:

```
report_cell [-connections]
```

Note:

The `report_cell -connections` command does not support physical-only cells.

Adding Power and Ground Rings

After you do floorplanning, you need to add power and ground rings.

Note:

If a blockage does not allow the tool to place the rings in the area you specified, it places the rings in the closest legal position.

To add power and ground rings,

1. Choose Preroute > Create Rectangular Rings.

The Create Rectangular Rings dialog box appears.

Alternatively, you can use the `create_rectangular_rings` command.

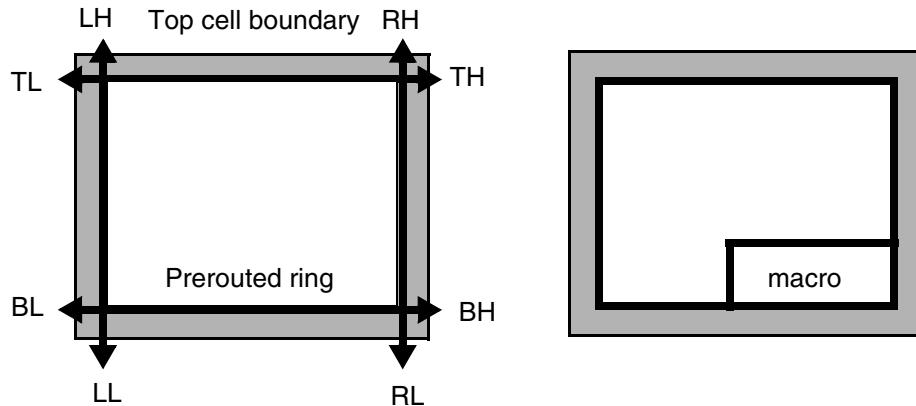
2. Specify the power ring characteristics for both power and ground.

- Nets – Enter the names of the nets for which you want to specify rings. To specify two nets, separate the names with commas.
- Side – Select the sides of the ring that you want to create. Select all sides of the ring to create a rectangular ring.
- Width – Enter the widths (left (L), right (R), bottom (B), and top (T)) of the respective sections of the ring.
- Layer – Enter the metal layer (left (L), right (R), bottom (B), and top (T)) to be used for the respective sections of the ring.
- Extend – You can extend any ring segment to the cell boundary or to the first target on the same net. To do so, enable the desired extension options, as defined in the following table.

Option	Extends
LL	The left segment of the ring low
LH	The left segment of the ring high
RL	The right segment of the ring low
RH	The right segment of the ring high
BL	The bottom segment of the ring low (left)
BH	The bottom segment of the ring high (right)
TL	The top segment of the ring low (left)
TH	The top segment of the ring high (right)

Selecting all the Extend options extends the ring from each corner to the top cell boundary in all directions.

Selecting LL and TH creates a macro ring as in this example.



- Offsets – Select an option for setting the ring offsets.

Option	Applies offsets
Absolute	Specified by you. This option applies the offsets without design rule checking adjustments. If the specified offset results in a DRC violation for a section of the ring, that section is omitted. It prevents automatic adjustment to correct violations. This is the default.
Applied after automatic adjustments	The ring segments are automatically adjusted to avoid any DRC violations.

Enter the offset for each side of the ring in the left, right, bottom, and top boxes.

- Create innermost core ring conservatively

Select this option if you want the innermost core ring placed far enough away from the core boundary to allow cell placement without design rule checking violations. You might want to use this option when you are creating core rings before cell placement. The tool scans all the standard cells to determine the maximum distance any cell can extend beyond the core boundary after placement. That maximum distance is placed between the innermost core ring and the core boundary.

- Ignore parallel targets

By default, each ring segment connects all passing targets. Select this option to skip those targets that are parallel to the segment.

- Extend for multiple connections

Select this option to extend ring segments to reach more targets on the same net. The extension continues until either there are no more targets in the respective directions, or the next target exceeds the spacing threshold set in the “for gap less than” box.

In the “for gap less than” text box, enter the space threshold for multiple connections. The default is 0.0.

- Use advanced via rules

Select this option to use the advanced via rules that have been previously set by the `set_preroute_advanced_via_rule` command. You can also Choose Preroute > Set Preroute Advanced Via Rule. ([“Setting Preroute Advanced Via Rules” on page 7-35](#).)

- Around

Select the area for which you want to specify rings.

Options	Creates rings around
Core	The predefined core.
Specified macros	The cell instances you specify by typing the instance names.
Specified as a group	The cell instances you specify by treating all the instances as one group.
Except macros	All macros in the design, except those specified in the instances field.
Region	A rectangle you specify by typing values for each coordinate. A rectangle is a collection of two points that specifies the lower-left and upper-right corners. A point is a collection of x- and y-coordinates. You can specify a region by drawing its outline in the layout (CEL) window. Use the left mouse button to enter the region points, and click the right mouse button to close the region. The tool will automatically pick up the coordinates.

3. Click OK.

Creating Rectilinear Rings

You can use the `create_rectilinear_rings` command to creates rectilinear rings around the core area or around macro cell instances in the design. The difference between this command and the `create_rectangular_rings` command is the ability to specify a list of macro cell instances that are excluded from the area bounded by the power and ground rings.

You can use the `create_rectilinear_rings` command options to control the location of the rings, and the width and layers of their segments.

The `create_rectilinear_rings` command uses the following syntax:

```
create_rectilinear_rings
[-nets {collection_of_nets}]
[-around core | macros | all_macros_except_specified]
[-macro_cells {collection_of_macro_cells}]
[-offset {x_offset y_offset}]
[-layers {h_segment_layer_name v_segment_layer_name}]
[-width {width_of_h_segments width_of_v_segments}]
[-space {space_between_h_segments space_between_v_segments}]
[-exclude_instances {collection_of_macro_cells}]
[-extension_of_excluded_instances {x_extension y_extension}]
[-max_deviation distance]
[-create_bridges {offset space}]
[-ignore_parallel_targets]
[-undo]
```

See the `create_rectilinear_rings` man page for information about the options.

Creating PG Rings Around a Group of Hard Macros

You can create power and ground rings around a group of hard macro blocks with either rectangular or rectilinear boundaries. Straps can also be inserted between the blocks if needed. As a prerequisite, you must first specify a power network synthesis region for the group of hard macro blocks by using the `set_fp_rail_region_constraints` command. The specified power network synthesis region is then shrunk, based on all the hard macro blocks inside the region, to provide a group block ring boundary, while still maintaining the region's topology. Once a power network synthesis region is defined, you can create the PG rings around the boundary of the hard macro group.

You can also preview the created group block PG ring results before committing the group block PG ring.

To create a block ring for a group of hard macro blocks,

1. Choose Preroute > Create Power Network Group Block Ring.

The Create Group Block Ring dialog box appears.

Alternatively, you can use the `create_fp_group_block_ring` command.

2. Specify the group block ring characteristics for both power and ground.

- Power Ground nets – You must specify the names of the power and ground nets on which to create the group block rings.
- Horizontal Ring Settings – Specify a horizontal metal layer on which to create the PG rings. The default is the top-most horizontal layer.

Spacing – Enter the horizontal ring spacing in microns between the power and ground nets. The default is the minimum spacing that is defined in the technology file.

Offset – Enter a horizontal ring offset in microns. The offset is measured from the block boundary. The default is the minimum spacing between the hard macro blocks and the ring that is defined in the technology file.

Width – Enter a horizontal ring width in microns. The default is the minimum width that is defined in the technology file.

- Vertical Ring Settings – Specify a vertical metal layer on which to create the PG rings. The default is the top-most vertical layer.

Spacing – Enter the vertical ring spacing in microns between the power and ground nets. The default is the minimum spacing that is defined in the technology file.

Offset – Enter a vertical ring offset in microns. The offset is measured from the block boundary. The default is the minimum spacing between the hard macro blocks and the ring that is defined in the technology file.

Width – Enter a vertical ring width in microns. The default is the minimum width that is defined in the technology file.

- Horizontal Strap Settings – Enter a horizontal strap layer and a horizontal strap width. The straps are inserted inside the narrow regions between neighboring hard macro blocks. The default is the top-most horizontal layer and the minimum width that is defined in the technology file.
- Vertical Strap Settings – Enter a vertical strap layer and a vertical strap width. The straps are inserted inside the narrow regions between neighboring hard macro blocks. The default is the top-most vertical layer and the minimum width that is defined in the technology file.
- Output directory – Specify the name of the directory in which to store results of the created block ring. The default is the `./pna_output` directory.

- Skip strap – Channels between the neighboring hard macro blocks are then identified to see where PG straps can be inserted, and any channels that overlap are merged. The straps are inserted in the middle of each channel between the neighboring blocks unless you specify the “Skip strap” option or the channel between the blocks is too narrow.
- Commit – Click the “Commit” button to generate a real power and ground group block ring and straps based on the results from the `create_fp_group_block_ring` command. Vias, along with wires, are also created at the intersection of two wires in opposite directions.

Alternatively, you can use the `commit_fp_group_block_ring` command.

3. Click OK or Apply.

Adding Power and Ground Straps

After you add the power and ground rings, you need to add the power and ground straps. The straps are automatically connected to the closest power and ground ring at, or beyond, both ends of the straps.

Note:

Using a few wide straps rather than many thin straps improves the placement quality and decreases the placement runtime.

To add power and ground straps,

1. Choose Preroute > Create Power Straps.

The Create Power Straps dialog box appears.

Alternatively, you can use the `create_power_straps` command.

2. Select the direction to specify that the strap starting position be horizontal or vertical. The default is horizontal.
3. Enter the names of the nets for which you want to specify straps. To specify two or more nets, separate the names with commas. The number of net names specified determines the number of straps in a group.

Next, enter the width you want for the straps, followed by the metal layer on which to create the straps.

4. Configure the placement of straps with one of the following options:

Options	Description
Groups & Step	Places straps based on the number of groups and the step or space, between groups you specify.
Groups & Stop	Places straps based on the number of groups and the stop point you specify. The tool determines the step distance.
Step & Stop	Places straps based on the step and the stop point you specify. The tool determines the number of groups.
Rows	Creates straps over standard cell areas; automatically determines the locations of straps.
Macros	Creates straps over pins of macro cells; automatically determines the locations of straps.

Note:

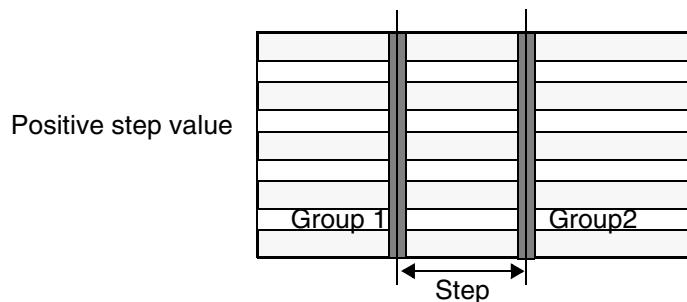
For vertical straps, a positive value places the next strap in the group to the right of the existing strap and a negative value places the next strap to the left of the existing strap. For horizontal straps, a positive value places the next strap above the existing strap and a negative value places the next strap below the existing strap.

- Groups

Enter the number of strap groups you want to create. When you select Step & Stop, this option is deactivated.

- Step

Enter the distance from the reference strap of one group to the reference strap of an adjacent group. When you select Groups & Stop, this option is deactivated.



- Stop

Enter the location where you want strap placement to stop. When you select Groups & Step, this option is deactivated.

- Pitch within Group

Enter the distance you want placed between the center lines of two adjacent straps within the same group.

A positive value places the next strap to the right of the first, if they run vertically, or up, if they run horizontally. A negative value places the next strap to the left, if they run vertically, or down, if they run horizontally.

- Low ends

Select an option for placement of low strap ends.

Option	IC Compiler places the low end of the strap...
At core boundary	At the core boundary
At first target	At the first target on the same net
At	At the coordinate you specify

Extend low – Extends the straps from the low end until the strap reaches the first target on the same net.

Extend to low boundaries and generate pins – If there is no such target, extends the low end of the strap to the boundary and generates a pin. The strap stops when it reaches the first target on the same net.

Force – Forcefully extends the low end of the strap to the boundary and generates a pin, even if the strap is connected to the first target.

- High ends

Select an option for placement of high strap ends.

Option	IC Compiler places the low end of the strap...
At core boundary	At the core boundary
At first target	At the first target on the same net
At	At the coordinate you specify

Extend high – Extends the straps from the high end until the strap reaches the first target on the same net.

Extend to high boundaries and generate pins – If there is no such target, extends the high end of the strap to the boundary and generates a pin. The strap stops when it reaches the first target on the same net.

Note:

The high end of a strap is the top end of a vertically oriented strap or the right end of a horizontally oriented strap. The low end of a strap is the bottom end of a vertically oriented strap or the left end of a horizontally oriented strap.

Force – Forcefully extends the high end of the strap to the boundary and generates a pin, even if the strap is connected to the first target.

- Use special via rule

Select this option if you want the tool to create a matrix of small vias instead of one large via at intersections, using the X and Y offset, size, and step values you specify. The use of smaller vias can free some routing space.

5. Click OK.

Additional options that you can select when adding power and ground straps are described in the following table.

Option	Description
Look inside std cells	Instructs the tool to create vias to connect straps to pins in standard cell pins.
Ignore cell boundary	Keeps out-of-bound straps.
Do not merge targets	Specifies that the target objects are not merged before the connections are created.
Ignore parallel targets	Ignores parallel targets. By default, the parallel targets are defined by their x and y dimensions. For example, when a target's x dimension is greater or equal to its Y dimension and you are creating horizontal straps, the target is not connected. The same is true for vertical straps.
Do not route over macros	Select to prevent the prerouter from generating wires over the macro cells. By default, this option is switched off. If you switch it on, the parts of a strap that go over a macro are removed.

Option	Description
Extend for multiple connections for gap	<p>Select to extend the low and high ends of the strap to connect more targets on the same net. The extension continues until either there are no more targets in the respective directions or the next target exceeds the spacing threshold set in the gap box.</p>
	<p>for gap less than – Enter the space threshold to be used for the extension process. Adjacent targets closer than this value are connected to each other.</p>
Marked as std cell pin connections	<p>When this option is selected, the created power and ground straps are marked as the “std conn” type instead of the “strap” type.</p>
Marked as rings	<p>When this option is selected, the created power and ground rings are marked as the “ring” type instead of the “strap” type.</p>
Distributed routing	<p>Enables distributed routing. Distributed routing divides the design into several routing partitions, based on size and congestion. These partitions are then routed on separate CPUs, in parallel on distributed machines, greatly reducing the overall runtime.</p>
	<p>Number of CPUs – Enter the number of CPUs that IC Compiler can use. The default is 1. For example, with four CPUs available, a typical runtime improvement would be a 3.5 times reduction in overall routing time.</p>
Keep floating pieces	<p>Keeps extra wire pieces that are not connected to other wires</p>
Clip at top cell boundary	<p>Cuts the top cell boundaries. This option is useful for creating straps within the boundaries of a rectilinear design.</p>
Optimize wire locations	<p>Specifies that the created straps might be adjusted by small distances horizontally or vertically so that more tracks are available later for the detail router.</p>
Special rules	<p>Use this field to reference the set of special rules previously specified by the <code>set_preroute_special_rules</code> command. See “Defining a Set of Special Preroute Rules” on page 7-34.</p>

Prerouting Standard Cells

You can connect power and ground pins in standard cells to the straps and rings of the power mesh and connect power and ground rails in the standard cells. To make sure the global router can recognize the routing obstruction, preroute the standard cells before performing global routing.

If the top cell does not have pads, you can use the “Extend to boundaries and generate pins” option in the Preroute Standard Cells dialog box. This option creates a power and ground extension wire to the cell boundary and generates a power and ground pin if the following conditions are met:

- The power and ground pin is not already connected to power and ground in the direction in which IC Compiler would create the extension wire.
- Creating the power and ground extension wire does not cause a design rule violation.

Pins are marked as fixed to prevent them from being moved by place and route operations.

To preroute standard cells,

1. Choose Preroute > Preroute Standard Cells.

The Preroute Standard Cells dialog box appears.

Alternatively, you can use the `preroute_standard_cells` command.

2. Click the Connection Mode tab.

- Select one of the following options:

Rail – Connects standard cells by rails. This is the default.

Tie – Connects standard cells by tying all the power and ground pins to nearby targets.

If you select this option, enter the maximum routing width. The default is 0.

Net – Connects pins of standard cells that are included with the specified net to nearby rings or straps.

If you choose the Net option, you can also specify a maximum fanout. This is the maximum number of pins that are connected to the ground rings or straps. The default is 10.

You can also set the routing layer and width for both vertical and horizontal segments

3. Click the Routing Options tab.

- Connect pins only on layer number – Select this option to connect pins on a specified metal layer. By default, any layer could be used.
- Connect – Select one or both of the following:

Vertically – Connects power and ground in the vertical direction.

Horizontally – Connects power and ground in the horizontal direction. This option is selected by default.

- Determine the pin width – You can determine the pin width by the width of the pin at its extreme edges or by the width of the most extended pin.

Additional options that you can select when connecting power and ground pins in the standard cells to the power and ground straps are described in the following table.

Option	Description
Extend for multiple connections	Extends the low and high ends of standard cell connections to reach more targets on the same net. The extension continues until either there are no more targets in the respective directions or the next target exceeds the specified spacing threshold.
Extend to boundaries and generate pins	Creates a power and ground extension wire to the cell boundary and generates a power and ground pin for the extension wire.
Keep floating rail segments	Keeps floating rail segments that are not connected to targets; otherwise, they are removed. The default is on.
Avoid merging vias	Prevents the merging of existing vias.
Optimize via locations to save tracks	Adjusts vias by small distances, horizontally or vertically, so that more tracks are available later for the detail router.
Use special via rule	Creates a matrix of small vias instead of one large via at intersections. Using smaller vias might free some routing space. You must specify the x- and y-offsets, size, and step.
	You can apply the x- and y-offsets to both the lower-left and upper-right corner of the wire intersection area by selecting the “offset both sides” option. By default, offsets are applied only to the lower-left corner.
Do not route over macros	Prevents the prerouter from generating wires over the macro cells. By default, this option is off. When it is on, the parts of a strap that go over a macro are removed.
Fill all empty rows	Fills rows without cells with power and ground rails.
Prevent connections outside the working area	Prevents standard cell connections from extending beyond the specified working area

Option	Description
Cut out rails between targets when there are no pins	Checks whether a rail (single wide metal segment) covers at least one power pin between the neighboring power straps or rings. If no standard cells pins are found, the rail is not created in that area of the row.
Distribute routing	Enables distributed routing. Distributed routing divides the design into several routing partitions, based on size and congestion. These partitions are then routed on separate CPUs in parallel on distributed machines, greatly reducing the overall runtime Enter the number of CPUs. The default is 1.

4. Click the Scope tab.

- Use top cell's bounding box – This option sets the working area to the top cell's bounding box.

Net names – Enter a list of net names in the text box. The net names in the list must be separated by commas.
- Use specified bounding box – You can select this option to specify a user-defined bounding box.

Coordinates – Enter the x- and y-coordinates in the Coordinates field, or select the Rectangle button and draw the rectangle in the layout view; or select the Rectilinear button and draw the rectilinear shape in the layout view.
- Filter Ports by name – Select from the following:
 - Filter off – Use this option when the selection of pins for routing is not dependent on their names.
 - Connect matched – Routes only those pins whose names are matched to at least one specified pattern.
 - Skip matched – Routes all pins except those pins whose names are matched to at least one specified pattern.
- Filter Ports of cells by master name – Select from the following:
 - Filter off – Use this option when the selection of pins for routing is not dependent on the names of their cell masters.
 - Connect matched – Routes only those pins whose cell master names are matched to at least one specified pattern.
 - Skip matched – Routes all pins except those pins whose cell master names are matched to at least one specified pattern.

- Filter Ports of cells by instance name – Select from the following:
 - Filter off – Use this option when the selection of pins for routing is not dependent on the names of their cell instances.
 - Connect matched – Routes only those pins whose cell instance names are matched to at least one specified pattern.
 - Skip matched – Routes all pins except those pins whose cell instance names are matched to at least one specified pattern.
 - Filter Ports of cells by voltage area located in– Select from the following:
 - Filter off – Use this option when the selection of pins for routing is not dependent on the name of the voltage area to which there cell instance is placed.
 - Connect matched – Routes only those pins whose cell instances are placed in a voltage area whose name is matched to at least one specified pattern.
 - Skip matched – Routes all pins except those pins whose cell instances are placed in a voltage area whose name is matched to at least one specified pattern.
5. Click the DRC button to open the Set Preroute DRC Options dialog box. (See “[Setting Preroute Design Rule Checking Options](#)” on page 7-32.)
 6. Click OK or Apply.

Setting Preroute Design Rule Checking Options

You can set preroute design rule checking (DRC) options to change the internal rules for design rule checking for the power and ground commands.

To change the internal rules for design rule checking,

1. Choose Preroute > Set Preroute DRC Options.

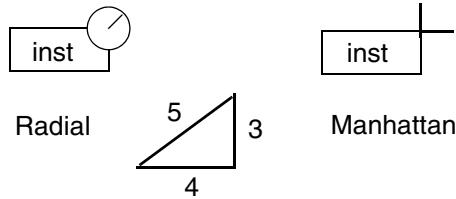
The Set Preroute DRC Options dialog box appears.

Alternatively, you can use the `set_preroute_drc_strategy` command.

2. Set the options, depending on your requirements.

- Spacing – Specify the mode for design rule checking.

Select “Radial” if your spacing rules are defined radially. A radial rule checks for spacing violations within a circular area. This is the default.



Select “Manhattan” if your spacing rules are defined as horizontal and vertical distances. A Manhattan rule checks for spacing violations within the defined distance, horizontally and vertically.

- Protect pin access edge – Select this option to prevent the access edge of the signal pin from being blocked by the prerouter. Select “Within pin layer switch” to protect the pin access edge within the double pin layer pitch distance. Select “Within range,” which is the default, to protect the pin access edge within the range you specify.
- Treat fat blockage as fat wire – Select this option to treat fat blockages as fat wires during DRC. Deselect the default to treat fat blockages as thin wires.
- Use fat via if wires meet width requirement –Select this option to use fat vias for non-turning layer changes when either wire meets the width requirement for fat vias.
- Ignore contents of standard cells – Select this option to ignore objects in standard cells during DRC.
- Ignore top level pins– Select this option to ignore top level pins during DRC.
- Ignore same net metal-metal – Select this option to ignore same-net metal-to-metal spacing during DRC.
- Quick check – Select this option to skip the final stage of DRC when a possible same-net metal-to-metal spacing error is found.
- Merge thin wires – Select this option to specify that the prerouter checks and merges thin wires into fat wires during DRC.
- Disable all DRC checking – Select this option to disable all design rule checking.
- Report failed connections information – Select this option to report detailed information about any failed standard cell connections that are found during DRC.
- Jogging – Select this option to define the maximum distance allowed for a dog-leg connection on the same layer (not used for layer switching). Typically, you should set a range that allows the prerouter to perform limited jogging, and only when necessary. The default is 0 (no jogging).
- Minimum metal layer – Select this option to set the minimum metal layer to be used for prerouting. The default is metal 1.

- Maximum metal layer – Select this option to set the maximum metal layer to be used for prerouting. The default is the highest metal layer in the current design.
- Honor shapes of nets – By default, if a net is not marked as power or ground, all of its routing shapes are ignored. If you want this routing to be considered, that is, DRC violations are to be avoided with these objects, select this option and enter a list of net names having such objects. Nets in the list must be separated by commas.

To specify all nets, select “All”. This is the default.

3. Click OK or Default.

Defining a Set of Special Preroute Rules

You can use the `set_preroute_special_rules` command to create a special set of preroute rules. The special rules that you define must be honored by the `create_power_straps` command and the `preroute_instances` command. The commands reference these special rules by `-name` argument that you specify.

The `set_preroute_special_rules` command uses the following syntax:

```
set_preroute_special_rules
  -name string
  [-do_not_connect_pins]
  [-extension_layers list_of_layers]
  [-extension_width width_of_segments]
  [-extension_space space_between_segments]
  [-extension_count {min_count max_count}]
  [-extension_via_array {columns_count rows_count}]
  [-extension_try_jog threshold]
  [-leave_space_for_io_connections]
  [-resolve_conflicts_by_jogs]
  [-connection_point_default {offset pitch}]
  [-connection_point_layers list_of_layers]
  [-connection_point_offsets list_of_offsets]
  [-connection_point_pitches list_of_pitches]
  [-first_strap_offset offset]
  [-strap_endcap distance]
  [-create_local_straps threshold]
  [-keep_straps_in_standard_cell_area]
  [-macro_cells {collection_of_macro_cells}]
  [-pin_layer layer_name]
  [-strap_to_pin_offset offset]
  [-end_point_offset distance]
  [-target_area {{dx-left dy-bottom} {dx-right dy-top}}]
  [-target_layer layer_name]
  [-jog_horizontal_layers list_of_layers]
  [-jog_vertical_layers list_of_layers]
  [-jog_width {width_of_h_segments width_of_v_segments}]
```

See the `set_preroute_special_rules` man page for information about the options.

Setting Preroute Advanced Via Rules

You can set preroute advanced via rules for creating via arrays at wire intersections. Sometimes creating a matrix of small vias instead of one large via at wire intersections is advantageous. Using smaller vias could free some routing space.

To set advanced via rules,

1. Choose Preroute > Set Preroute Advanced Via Rule.

The Set Preroute Advanced Via Rule dialog box appears.

Alternatively, you can use the `set_preroute_advanced_via_rule` command.

2. Set the options, depending on your requirements.

- Move via to center – Creates one via array at the intersection point where orthogonal segments connect, and places the array as close as possible to the intersection point.
- Apply offset to – You can apply the X and Y offsets to both the lower-left and upper-right sides of the wire intersection area. The default applies them only to the lower-left side.
- Only when recommended – Specifies that the router can change the offset values to avoid DRC violations with existing objects. If DRC violations cannot be avoided, the via is not created.
- X/Y step – Enter the maximum x- and y-distance allowed between the lower-left corners of adjacent via arrays. The default is 0.0.
- Apply X/Y offsets – Enter the offset from the lower-left corner of the wire intersection area at which the first via array will be created. The default is 0.0.
- Via rule – Set the following options to specify via rules on a per layer basis:
 - All via layers – When you select this option, it applies the default rules or sets an initial pattern to be used for setting rules for all non-specified via layers.
 - Cut via layers – When you select a cut layer, a via layer name or number (contact code) is selected automatically, depending on the wire width.
 - Via contact code – When you select a contact code (contact codes correspond to a cut layer), only this contact is used for the layer, and the wire width is disregarded.
- Size by – You can specify the width and height of the via array area, or you can set the size of an array by its dimensions instead of its real size.

Via area – Enter the width and height of the via array (the default).The number you specify is the X or Y length in user units.

Array dimensions – Enter the number of cuts for the via array in both the X and Y dimensions.

- Rotation mode – Specify a via rotation mode.

On – Allows the router to rotate a via in the preferred routing direction of the metal layer if doing so might save routing resources (the default).

Required – Forces the router to rotate a via.

Off – Prohibits the router from rotating a via.

3. Click OK or Apply.

Creating Preroute Vias

You can create vias between specified layers on either power or ground nets, on specified nets, or on specified bus nets.

To create preroute vias between specified layers,

1. Choose Preroute > Create Preroute Vias.

The Create Preroute Vias dialog box appears.

Alternatively, you can use the `create_preroute_vias` command.

2. Select an option to determine which nets to route.

Option	Description
Power and Ground	Routes power and ground nets. (default)
Specified nets	Routes only the nets you specify. If you select this option, you need to type the names of the nets in the text box.
Specified buses	Routes only the bus nets you specify. If you select this option, you need to type the names of the buses in the text box.

3. Specify the source layer from which the vias are dropped. The default drops vias from the first vertical layer.

4. Select the type of power and ground objects from where the vias are dropped.

Option	IC Compiler marks the object as a ...
Strap	strap
User	user-defined route
Std pin Conn	standard cell pin connection
Std Pin	standard cell pin
Macro/I/O pin	macro or I/O pin
Macro/I/O Pin Conn	macro or I/O pin connections
Ring	ring
Bus	bus

5. Specify the target layer to which the vias are dropped. The default targets the first horizontal layer.

- Connect to targets on all intermediate layers

When the option is selected, the vias are dropped from the source layer to all layers between the source and target layers. Otherwise, the vias are dropped to the target layer only.

- Ignore parallel targets

When this option is selected, there are no vias created between two parallel objects.

- Do not merge targets

When this option is selected, target object are not merged before the vias are created.

6. Select the type of power and ground objects to which the vias are dropped.

7. Select the “Special via rule” option if you want the tool to create a matrix of small vias instead of one large via at wire intersections. Using smaller vias could free some routing space. You must specify the X and Y Offsets, Size, and Step.

- X/Y Offset

Enter the offset from the lower left corner of the wire intersection area where you want to create the first special via.

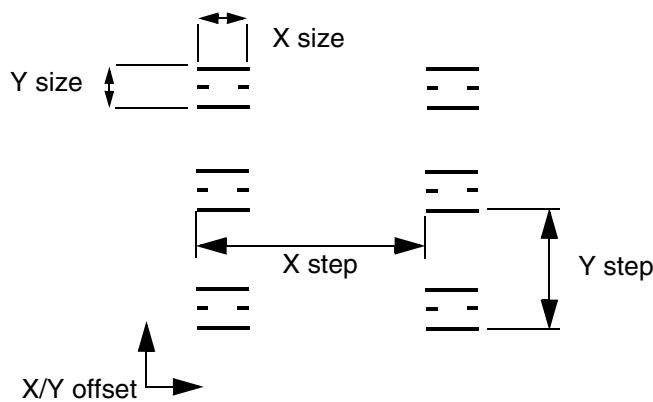
- X/Y Size

Enter the size of the special vias. The number you type is the x or y length in your units.

- **X/Y Step**

Enter the x and y distance you want between the special vias. When there are design rule checking violations, some vias might not be created.

For example, if you want to cut-up the intersection area where one single via would have been dropped into six sub-areas, the X/Y steps would be as shown below:



- **Offset both sides**

Select this option to apply the X and Y offsets to both the lower-left and upper-right corner of the wire intersection area. The default applies them only to the lower-left corner.

8. Specify the rectangular area within which to create the vias.

You can click on the rectangle button and draw an area, or you can type the coordinates in which you want to preroute the standard cells. By default, the working area extends across the entire top cell, from the origin (0,0), to the upper right corner of the top cell. Standard cells that fall within that area or touch the boundary are routed.

9. Adjust the via locations.

Select the “Optimize via locations” option to adjust the vias by small distances horizontally or vertically so that more tracks are saved for the detail route. By default this option is deselected.

10. Mark created vias with different attribute types

Select the “Mark Vias attribute as” option to mark the created vias with different types of via attributes. You can select the following via attributes from the pull-down menu.

```
strap  
user_defined  
standard_cell_pin_connection  
macro_io_pin_connection  
ring  
bus
```

The default is the same type as the source object of the via. You can use this option to override the default via type.

11. Specify advanced via rules.

Select this “Use preset advanced via rules” option to specify advanced via rules that have been previously set. By default, this option is deselected. (See [“Setting Preroute Advanced Via Rules” on page 7-35](#).)

Performing Low-Power Planning for Multithreshold-CMOS Designs

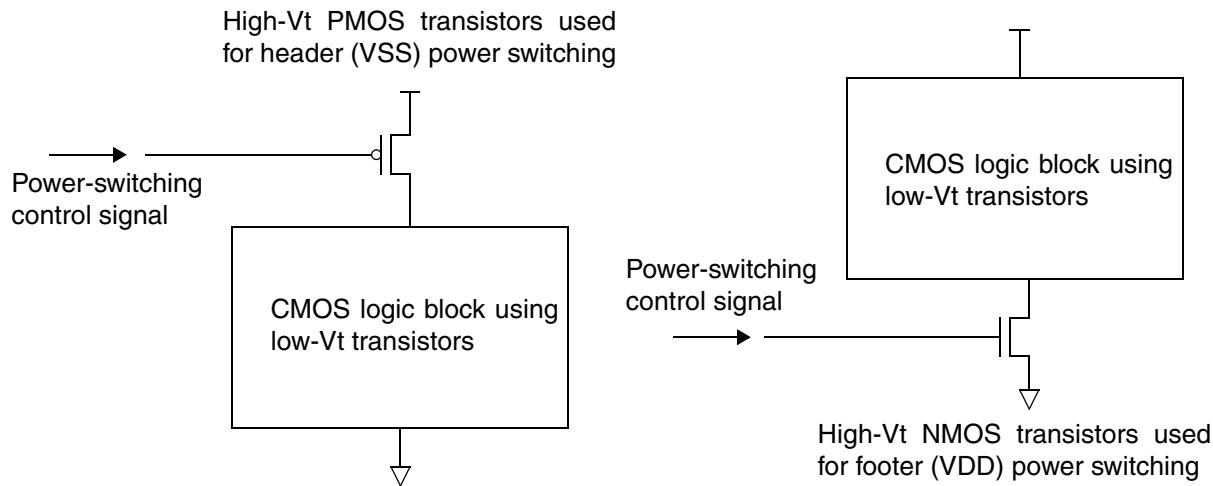
You can perform floorplanning for low-power designs by employing power switching. Power switching has the potential to reduce overall power consumption substantially because it lowers leakage power as well as switching power. It also introduces some additional challenges, including the need for a power controller, a power-switching network, isolation cells, and retention registers.

A block that can be powered down must receive its power through a power-switching network, consisting of a large number of transistors with source-to-drain connections between the always-on power supply rail and the power pins of the cells. The power switches are distributed physically around or within the block. When the network is switched on, it connects the power to the logic gates in the block. When switched off, the power supply is effectively disconnected from the logic gates in the block.

High threshold voltage transistors from multithreshold-CMOS technology are used for the power switches because they minimize leakage and their switching speed is not critical. The power switches turn off the connection to power or ground for low-voltage standard cells and thereby turn off leakage power when the design is in standby mode.

Power is turned off to a given portion of a design by the use of header and footer switches implemented with PMOS and NMOS transistors, respectively. PMOS header switches can be placed between VDD and the block power supply pins, or NMOS footer switches can be placed between VSS and the block ground pins, as shown in [Figure 7-8](#). The number, drive strength, and placement of switches should be chosen to give in an acceptable voltage drop during peak power usage in the block.

Figure 7-8 Power-Switching Network Transistors



The switching strategy shown in [Figure 7-8](#) is called a coarse-grain strategy because the power switching is applied to the whole block. Multiple transistors in parallel drive a common supply net for the block. In a fine-grain strategy, each library cell has its own power switch, allowing fine-grain control over which cells are powered down. The fine-grain approach has better potential for power savings, but requires significantly more area.

This section includes the following topics:

- [Performing Multithreshold-CMOS Exploration Using Power Network Analysis](#)
- [Inserting Power-Switching Cells for Multithreshold-CMOS Designs](#)
- [Creating Virtual Power and Ground Nets and Connecting the Power-Switching Cells](#)
- [Optimizing the Power-Switching Cells to Meet IR Drop Targets](#)
- [Performing Multithreshold-CMOS Ring Placement](#)
- [Performing Power Network Analysis on Power-Switching Cells in Multithreshold-CMOS Designs](#)
- [Automatically Connecting Switch Pins](#)
- [Advanced Strategy to Control Aligned Permanent Power Straps](#)

Performing Multithreshold-CMOS Exploration Using Power Network Analysis

You can explore and estimate possible placement combinations of multithreshold-CMOS header or footer cells for power gate insertion and quickly obtain an IR drop result before power planning; replace existing header or footer cells and perform IR drop analysis; and perform header or footer array sizing to meet the IR drop targets.

Exploring the Placement of Multithreshold-CMOS Power Switching Cells in the Power Network

In low-power multithreshold-CMOS designs, power-switching cells, also called header and footer cells, require special placement near power straps. Use the `explore_power_switch` command and options to explore and estimate the placement of header or footer cells early in the floorplanning stage before the power network is created.

Prerequisites: You must perform the following steps before you can run the `explore_power_switch` command. These steps are necessary for the power network analysis engine to compute the IR drop.

1. Set the library cell attributes `mtcmos_resistance` and `mtcmos_pin_layers` to specify the power-on resistance values and power-pin metal layers for all multithreshold-CMOS power switches (header and footer cells).

To set these attributes, use the `set_attribute` command. These attribute values are stored by creating properties on the master cell in the top design. Power network analysis can recognize multiple resistance and pin layer attributes for different types of power-switching cells.

2. Define the virtual power or ground pad for power network synthesis or power network analysis by using the `create_fp_virtual_pad` command.
3. Verify that the virtual and real power and ground nets of all standard cells are properly connected logically in the netlist by using the `derive_pg_connection` command. A standard cell to be powered down should connect to a virtual power or ground net. A standard cell that is always on should connect to a real power or ground net.

Note:

The `explore_power_switch` command assumes that the metal preroutes of the power network do not exist. If the power straps already exist, use the `create_power_switch_array` command to insert the multithreshold-CMOS header and footer cells.

The `explore_power_switch` command uses the following syntax:

```
explore_power_switch
-lib_cells lib_cells_or_power_switches
```

```

-virtual_pg_net virtual_pg_net_name
-real_pg_net real_pg_net_name
[-header]
[-bounding_box bounding_box]
[-x_increment x_inc_list]
[-y_increment Y_inc_list]
[-orientation {N | W | S | E | FN | FE | FS | FW}]
[-voltage_area voltage_area]
[-skip_placement]
[-legalize_placement]
[-skip_pns]
[-run_pns_script file_name]
[-design hierarchy_name]
[-save_placement prefix]
[-preroute_mode_real_pg_port {rail | tie | net}]
[-skip_preroute_real_pg_port]
[-preroute_mode_virtual_pg_port {rail | tie | net}]
[-skip_preroute_virtual_pg_port]

```

You must use the `-lib_cells` option to specify a list of header or footer library reference cells for use during the exploration phase. You must also specify a two-dimensional grid on which the header or footer cells are automatically inserted. The `explore_power_switch` command first runs a fast placement of the existing standard cells and macros in the design. Then, based on the input list you specify, it explores up to a maximum of 27 possible combinations of header or footer cell placements with different x- and y-spacings between the cells.

By default, the `explore_power_switch` command does not modify the current design. If you want to save a header or footer cell placement generated during the explorations, use the `-save_placement` option.

For each placement of the header or footer cells, the `explore_power_switch` command estimates the IR drop by using the power network analysis engine and reports the results. Based on those results, you can select the appropriate header or footer cells and their placement pitch to achieve the desired IR drop for your design.

Inserting Power-Switching Cells for Multithreshold-CMOS Designs

In low-power multithreshold-CMOS designs, power-switching cells, also called header and footer cells, require special placement. You can use the `create_power_switch_array` command to specify a 2-D insertion grid on which the header and footer cells are automatically inserted and placed in the logical netlist as well as the floorplan. The grid is a lithography grid.

The `create_power_switch_array` command uses the following syntax:

```

create_power_switch_array
-lib_cell lib_cell_or_power_switch
[-voltage_area voltage_area]

```

```

[-bounding_box rectangle]
[-relative_to_voltage_area]
[-design hierarchy_name]
-x_increment dx -y_increment dy
[-start_row index]
[-start_column index]
[-snap_to_row_and_tile]
[-orientation orientation]
[-respect string]
[-pattern normal |staggered]
[-prefix prefix_name]

```

The command honors voltage areas when the `-voltage_area` option is turned on. This option allows you to place the header and footer cells only in certain voltage areas. The default places them at the top level. If you turn on the `-snap_to_row_and_tile` option, the grid is automatically adjusted so that the header and footer cell placement is aligned on the appropriate cell row sites.

The inserted header and footer cells are not legally placed. If legalization is necessary (legalization works only on standard cells), you must specify a grid that is legal or you must run the `legalizer` (`legalize_placement` command) after running the `create_power_switch_array` command, to ensure that the cells are placed in legal locations.

Replacing Existing Header or Footer Cells

Use the `replace_power_switch` command to replace existing header or footer cell instances with a specified header or footer library cell of a larger or smaller size in one vertical or horizontal array.

Use this command during the multivoltage power planning stage to optimize the IR drop when the multithreshold-CMOS cells and power network physically exist in the design. An IR drop report is generated before and after cell replacement.

Prerequisites: You must perform the following steps before you can run the `replace_power_switch` command. These steps are necessary for the power network analysis engine to compute the IR drop.

1. Set the library cell attributes `mtcmos_resistance` and `mtcmos_pin_layers` to specify the power-on resistance values and power-pin metal layers for all multithreshold-CMOS power switches (header and footer cells).

To set these attributes, use the `set_attribute` command. These attribute values are stored by creating properties on the master cell in the top design. Power network analysis can recognize multiple resistance and pin layer attributes for different types of power-switching cells.

2. Define the virtual power or ground pad for power network synthesis or power network analysis by using the `create_fp_virtual_pad` command.

3. Verify that the virtual and real power and ground nets of all standard cells are properly connected logically in the netlist by using the `derive_pg_connection` command. A standard cell to be powered down should connect to a virtual power or ground net. A standard cell that is always on should connect to a real power or ground net.

The `replace_power_switch` command uses the following syntax:

```
replace_power_switch
-cells [instances]
-lib_cell [lib_cell_or_power_switch]
[-virtual_pg_net virtual_net]
[-real_pg_net real_net]
[-skip_analyze_power]
```

Use the `-cells` option to specify the existing header or footer cell instances that you want to replace, and use the `-lib_cell` option to specify the library cell to be used for replacement. The `-lib_cell` argument can contain only one footer or header library cell.

IR drop analysis is performed on virtual power or ground nets by using the power network analysis engine. You can specify a virtual power or ground net by using the `-virtual_pg_net` option. A virtual power net can be powered down by the header and footer cells to reduce leakage power.

The header or footer cell instances to be replaced should be connected to specified virtual and real power and ground nets. When you run the `replace_power_switch` command, the IC Complier tool adjusts the location of the incoming cells so that the metal connections of the pins are preserved.

The `replace_power_switch` command assumes that the header and footer cell instances are functionally equivalent to the specified library cell, have the same cell height (for a vertical array) or cell width (for a horizontal array), and that the locations of the pins are compatible. Otherwise, the cell replacement can result in power and ground connections that are physically broken, causing incorrect IR drop analysis.

Creating Virtual Power and Ground Nets and Connecting the Power-Switching Cells

You can use the `connect_virtual_pg_net` command to create virtual power and ground nets and connect the multithreshold-CMOS power-switching cells (header and footer cells) to those nets.

You must use the `-bounding_box` option to specify the rectangle containing the multithreshold-CMOS power-switching cells that are to be considered for connection and must use the `-net_type` option to indicate the type of nets (power or ground) to be created.

If the bounding box does not intersect a specified voltage area (`-voltage_area` option), no cells will be connected; in other words, only the header and footer cells that reside in the bounding box and in the defined voltage area will be connected to the virtual power and ground nets. If you do not specify a voltage area, only the cells in the top level and in the bounding box will be connected.

Note:

The connection of the power and ground nets is done only among the power-switching cells and does not involve the standard cells that are powered by the power-switching cells.

Optimizing the Power-Switching Cells to Meet IR Drop Targets

You can use the `optimize_power_switch` command to resolve IR drop problems by automatically sizing up or down the multithreshold-CMOS power-switching cells in the design.

Prerequisites: You must perform the following steps before you can run the `optimize_power_switch` command. These steps are necessary for the power network analysis engine to compute the IR drop.

1. Set the library cell attributes `mtcmos_resistance` and `mtcmos_pin_layers` to specify the power-on resistance values and power-pin metal layers for all multithreshold-CMOS power-switch cells (header and footer cells).

To set these attributes, use the `set_attribute` command. These attribute values are stored by creating properties on the master cell in the top design. Power network analysis can recognize multiple resistance and pin layer attributes for different types of power-switching cells.

2. Define the virtual power or ground pad for power network synthesis or power network analysis by using the `create_fp_virtual_pad` command.
3. Verify that the virtual and real power and ground nets of all standard cells are properly connected logically in the netlist by using the `derive_pg_connection` command. A standard cell to be powered down should connect to a virtual power or ground net. A standard cell that is always on should connect to a real power or ground net.

The `optimize_power_switch` command uses the following syntax:

```
optimize_power_switch
-real_pg_net real_pg_net
-virtual_pg_net virtual_pg_net
[-lib_cells lib_cells_or_power_switches]
[-cells instances]
[-legalize_placement]
[-remove_preroute]
```

```
[-preroute_mode_real_pg_port rail | tie | net]
[-skip_preroute_real_pg_port]
```

The `optimize_power_switch` command optimizes the target IR drop in the design. It sizes up all the multithreshold-CMOS header and footer cells to a maximum size needed to meet the IR drop target; it sizes down the header and footer cells when the IR drop is much less than the target you have specified.

Performing Multithreshold-CMOS Ring Placement

You can use the `create_power_switch_ring` command to create a full ring or a partial ring of multithreshold-CMOS power-switch cells around voltage areas or macro cells. The ring consists of multithreshold-CMOS power-switch cells with optional filler cells and corner cells.

You must specify the names of the library cells or power-switch cells by using the `-switch_lib_cell` option. Library cells should be used for non-UPF mode and power-switch cells should be used for UPF mode. You can use the `-density` option to control the number of power-switch cells that are inserted.

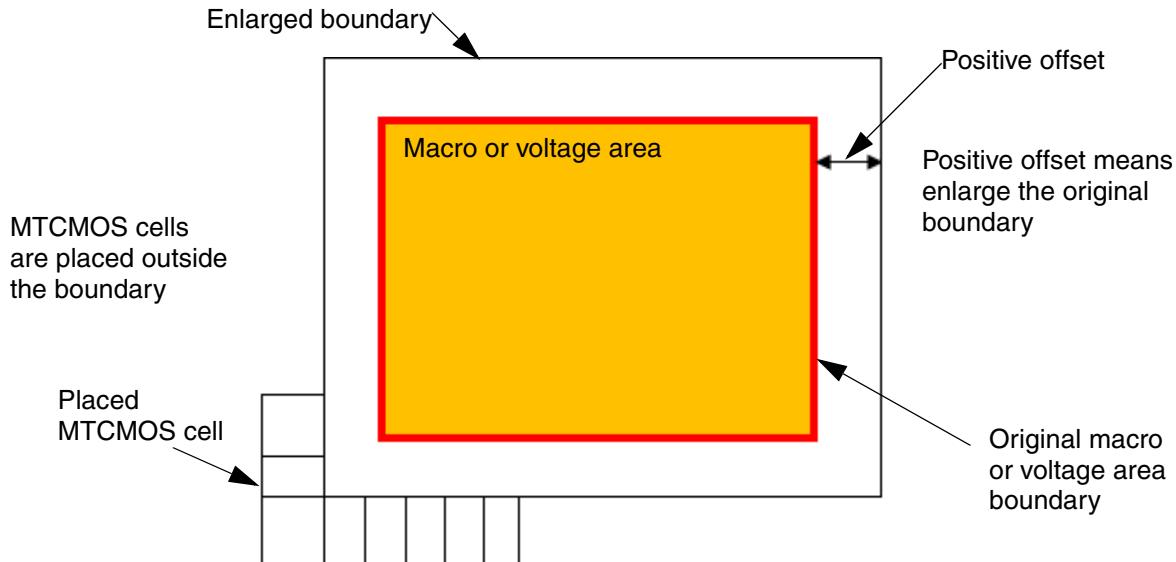
You can use the `-filler_lib_cells` option to specify a list of filler cells to be inserted between two power-switch cells or corner cells. The command will then try to fill any empty spaces to ensure that there are gaps in the ring. You can also use the `-outer_corner_lib_cell` option or the `-inner_corner_lib_cell` option to place corner cells at the outer (convex) corner or inner (concave) corner of the power-switch cell ring.

You can define a ring segment along boundary edges by using the `-start_point` and `-end_point` options. The command will create the ring segment in the counterclockwise direction.

By default, the command places the power-switch cell ring on the outside of the voltage area or macro cell boundary. You can use the `-offset` option to define the distance in which to enlarge or shrink the boundary of the ring with respect to the voltage area or macro cell boundary. You can set a positive offset value to enlarge the original boundary macro or voltage area boundary or you can set a negative offset value to shrink the boundary. A positive offset value places the power-switch cell ring inside a voltage area and outside a macro.

[Figure 7-9 on page 7-47](#) shows an example of ring placement using the `-offset` Option.

Figure 7-9 Multithreshold-CMOS Ring Placement -offset Option



The `create_power_switch_ring` command uses the following syntax:

```
create_power_switch_ring
  -area_obj voltage_area_or_macro
  -switch_lib_cell lib_cells_or_power_switches
  [-filler_lib_cells collection_of_filler_lib_cell_names]
  [-outer_corner_lib_cell outer_corner_lib_cell_name]
  [-inner_corner_lib_cell inner_corner_lib_cell_name]
  [-offset -offset f or -offset {f1 f2...}]
  [-design hierarchy_name]
  [-density]
  [-start_point {x y}]
  [-end_point {x y}]
  [-switch_orientation {N | W | S | E | FN | FE | FS | FW}]
  [-filler_orientation {N | W | S | E | FN | FE | FS | FW}]
  [-outer_corner_orientation {N | W | S | E | FN | FE | FS | FW}]
  [-inner_corner_orientation {N | W | S | E | FN | FE | FS | FW}]
  [-prefix]
  [-check_overlap]
```

Performing Power Network Analysis on Power-Switching Cells in Multithreshold-CMOS Designs

You can perform power network analysis on power-switching cells in multithreshold-CMOS designs.

However, before you can perform power network analysis on multithreshold-CMOS designs, you must do the following:

1. Specify the virtual-power-net, real-power-net pin name, and the layer of the multithreshold-CMOS cell.

For example, to specify a virtual-power-net pin VDDV located at metal1 and a real power net pin VDD located at metal6 for the multithreshold-CMOS cell, set the following environment variable:

```
icc_shell> set_attribute [get_physical_lib_cells "GSWITCH"] \
    "mtcmos_pin_layers" "VDD metal6 VDDV metal1"
```

2. Specify the power cell resistance between the virtual-power-net pin and the power-net pin. For example, to specify the resistance between virtual-power-net pin VDD1 and power-net-pin of multithreshold-CMOS switch cell “GSWITCH” as 30 ohm, enter the following command:

```
icc_shell> set_attribute [get_physical_lib_cells "GSWITCH"] \
    "mtcmos_resistance" 30
```

3. Describe the power net connectivity through gate cells by specifying equivalent nets and then run power network analysis. Power network analysis uses the “+” notation to recognize the equivalent nets.

When the power switch cell is turned on, to specify real-power-net VDD plus virtual-power-net VDD1 in the Power Ground nets text box in the Power Network Analysis dialog box, enter

VDD+VDD1

On the command line, enter

```
analyze_fp_rail -nets {VDD+VDD1}
```

Power network analysis analyzes the power network for both the real VDD power net and the virtual VDD1 power net through the multithreshold-CMOS cell GSWITCH connection.

Automatically Connecting Switch Pins

You can automatically connect switch pins of coarse-grain multithreshold-CMOS power switch cells to the control pin by using the `connect_power_switch` command.

Note:

This command requires that the `switch_cell_type` attribute be defined on the power switch cells in the logical library.

The syntax is

```
connect_power_switch -source object -port_name port_name
-mode hfn | daisy [-ack_out object] [-ack_port_name port_name]
[-direction horizontal | vertical] [-verbose] [-voltage_area list]
[-object_list objects] [-lib_pin library_pin_names]
```

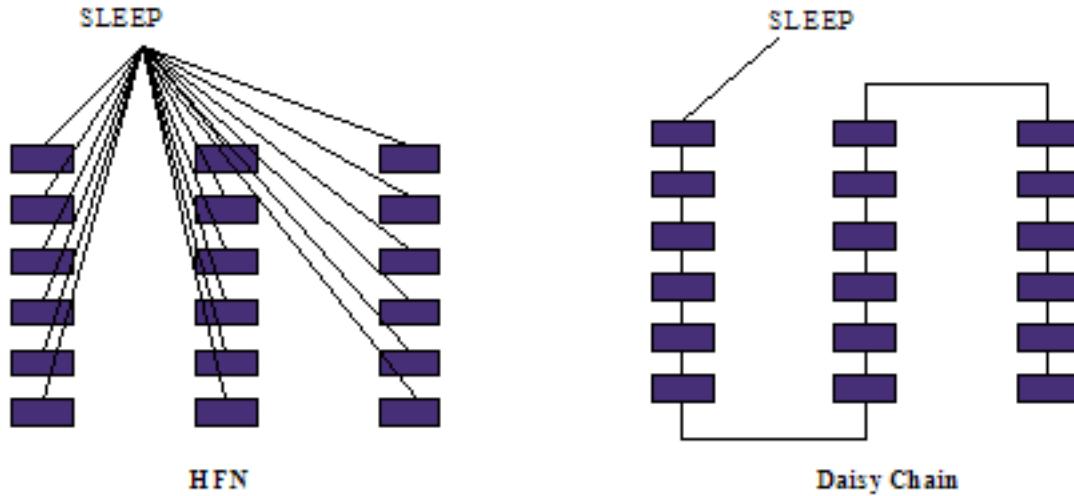
This command supports the following two ways of connecting the switch pin to the control pin:

- High-fanout (hfn) mode in which all switch cells are controlled in parallel by a single signal.
- Daisy chain in which switches are connected in a long chain; an acknowledge -out signal from one switch controls the next switch cell and so on. These cells are placed in a matrix pattern on the site rows.

Use the `-mode` option to choose the desired way of connecting the switch pin to the control pin.

[Figure 7-10 on page 7-49](#) shows how the switch pins are connected in each of these modes.

Figure 7-10 Modes for Connecting Switch Pin of MTCMOS Power Switch Cells to the Control Pin



At a minimum, you must specify the port or pin that drives the control signal (`-source` option), the base name to use for hierarchical ports created for the control signal connections (`-port_name` option), and the connection mode (`-mode hfn | daisy`). For the daisy-chain mode, you must also specify the port or pin connected to the acknowledge-out signal (`-ack_out` option) and the base name for hierarchical ports created for the acknowledge-out signal connections (`-ack_port_name` option).

By default, all switch pins are connected to the specified control signal. You can limit the connections to those within a specified set of voltage areas by using the `-voltage_area` option.

For high-fanout mode, you can limit the set of power switch cells connected to the control signal by using the `-object_list` option.

For daisy-chain mode, you can specify either the order in which to connect the power switch cells (`-object_list` option) or the direction in which to connect the power switch cells (`-direction` option). If your power switch cells have multiple switch or acknowledge-out pins, you must use the `-lib_pin` option to specify which pin to use for the automatic connections.

Performing Power Network Synthesis On Multivoltage Designs

Multivoltage power network synthesis in IC Compiler allows you to:

- Synthesize power nets and a common ground net on multiple voltage areas within a single power network synthesis run based on power plan constraints.
- Automatically allocate power budgets to each voltage area out of the total power budget area.
- Generate multi-net core rings in the top-level design and connect those core rings to corresponding power straps or voltage area rings inside each voltage area.
- Automatically align the power straps with multithreshold-CMOS cells in the voltage area.
- Commit all the synthesized power and ground nets simultaneously.

This section includes the following topics:

- [Performing Power Network Synthesis for Power-Gating Designs With Multithreshold-CMOS Switch Cells](#)
- [Defining Multivoltage Power Network Synthesis Constraints in a Specified Voltage Area](#)
- [Removing Voltage Area Constraints](#)
- [Reporting Voltage Area Constraints](#)
- [Allocating Power Budgets to Each Voltage Area](#)
- [Synthesizing the Voltage Areas](#)
- [Generating Multi-Net Core Rings and Connecting Power Straps](#)
- [Supporting Incomplete Power Network Analysis](#)
- [Committing the Synthesized Results](#)

Performing Power Network Synthesis for Power-Gating Designs With Multithreshold-CMOS Switch Cells

You can run power network synthesis concurrently on permanent and virtual power nets in power-gated voltage areas where multithreshold-CMOS power switch cells, also called sleep transistors, have been inserted in the power-down voltage areas. The synthesized power network should be based on an accurate IR drop simulation performed in a network consisting of the permanent and virtual power networks connected by the multithreshold-CMOS switch cells. Power is delivered from the power pads to the permanent network and then through switch cells to the virtual power network. Therefore, you should make connections between the permanent and virtual straps and their corresponding pins on the power switch cells or generate a virtual rail for the virtual power net.

This section includes the following topics:

- [Synthesizing the Power Network for a Power-Down Voltage Area](#)
- [Performing Automatic Power Switch Cell Synthesis for Power-Down Voltage Areas](#)
- [Specifying Different Multithreshold-CMOS Placement Styles](#)
- [Simulating the IR Drop Across the Multithreshold-CMOS switch Cells](#)
- [Synthesizing the Common Power Network](#)
- [Advanced Strategy to Control Aligned Permanent Power Straps](#)

Synthesizing the Power Network for a Power-Down Voltage Area

To synthesize the power network for a power-down voltage area, you must specify the name of the permanent net followed by the name of the virtual net.

The `set_fp_rail_voltage_area_constraints` command uses the following syntax:

```
set_fp_rail_voltage_area_constraints -voltage_area voltage_area_name
                                     -nets "permanent_net + virtual_net" -voltage_supply supply_voltage
```

For example,

```
set_fp_rail_voltage_area_constraints -nets "VDD + VDDL"-voltage_supply 1.2
```

Note:

If the Unified Power Format (UPF) commands define the power-down voltage area, power network synthesis automatically retrieves the permanent and virtual net information from the UPF version. You do not need to specify the `-nets` and `-voltage_supply` options.

Performing Automatic Power Switch Cell Synthesis for Power-Down Voltage Areas

You can perform automatic power switch cell synthesis for power-down voltage areas in IC Compiler power network synthesis. This can help you decide the optimal locations and numbers in which to insert the power switch cells together with the optimal wire size and pitch for both permanent and virtual power straps that meet the target IR drop at the design planning stage. The power switch cell type can be either a hard macro or a standard cell. After you run power network synthesis, you can display the synthesized power switch cells in an IR drop map. The location of synthesized power switch cells are highlighted, and the coordinates of each power switch cell are output to an ASCII file. Later, you can physically insert the power switch cells based on the coordinates in this power network analysis output directory (pna_output).

Note:

Currently, automatic power switch cell synthesis works only for header cells.

To perform automatic power switch cell synthesis for power-down voltage areas, follow these steps:

1. Specify the voltage area synthesis constraints.

Before you synthesize the switch cells, first specify the voltage area synthesis constraints by using the `set_fp_rail_voltage_area_constraints` command.

- Select the `-voltage_area voltage_area_names` option and specify the names of the voltage areas.
- Select the `-nets pg_nets` option and specify the name of the permanent net, followed by the name of the virtual net for the power-down voltage area.

```
-nets permanent_net + virtual_net
```

Note:

If the Unified Power Format (UPF) commands define the power-down voltage area, power network synthesis automatically retrieves the permanent and virtual net information from the UPF version. You do not need to specify the `-nets` option.

2. Specify the layer constraints for the power straps of the permanent and virtual power nets, followed by the net type on the layer in the power-down voltage area.

Specify the layer constraints for the power straps of both the permanent and virtual power nets, and the net type on the layer in the power-down voltage area by using the `set_fp_rail_voltage_area_constraints` command.

- Select the `-layer pg_layer` option, and specify the metal layer for the power and ground straps of the permanent and virtual power nets. Specify the maximum or minimum width of the power and ground straps in the voltage area by using the `-max_width` or `-min_width` options.

A multilayer power and ground structure are assumed to be in the power-down voltage area with permanent power nets on the top two layers and virtual power nets on other, usually lower, layers. To avoid DRC issues, permanent and virtual power nets should not be created on the same layer.

- Select the `-mtcmos_type permanent | virtual` option and specify the multithreshold-CMOS net type on the layer in power switch array synthesis. Only one power net type, either permanent or virtual, can be synthesized on one layer during power switch array synthesis. By default, the permanent net type is used for each layer.

3. Specify the power switch cell on-resistance and master cell name.

Specify the attributes of the multithreshold-CMOS power switch cell on-resistance, the master cell name and pin layers of its primary and secondary power pins.

The command syntax is:

```
set_attribute [get_physical_lib_cells switching_cell_name]
"mtcmos_resistance" resistance_value

set_attribute [get_physical_lib_cells switching_cell_name]
"mtcmos_pin_layers" "pin1_name pin1_layer pin2_name pin2_layer"
```

If there are multiple power switch cells in your design, you must also specify the library cell name to be used as the power switch in the voltage area by using the `-power_switch` option.

The command syntax is:

```
set_fp_rail_voltage_area_constraints -power_switch
lib_cell_or_power_switch
```

In non-UPF mode, use the cell reference (master) library name for the power switch cell name. In UPF mode, the power switch cell name defined in UPF is used.

4. Synthesize the power switch cells.

Synthesize the power switch cells and the power network simultaneously for the power-down voltage area.

Note:

You can synthesize the switch cells for only one voltage area at a time. If you specify multiple voltage area names, power network synthesis will issue an error message and exit.

The command syntax is:

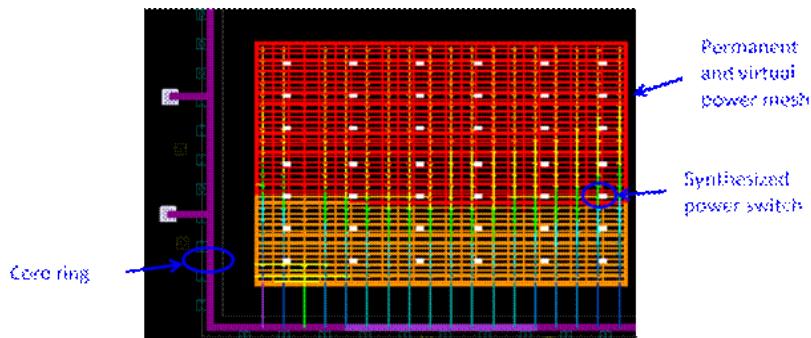
```
synthesize_fp_rail -synthesize_power_switch_array -voltage_areas
voltage_areas
```

- Use the `-synthesize_power_switch_array` option to perform power switch synthesis on the power-down voltage areas.

- Use the `-voltage_areas voltage_areas` option to specify the name of the voltage area that you want to synthesize.
5. Display the IR drop map result of the power network.

After you run power network synthesis using the `synthesize_fp_rail` command, you can display an IR drop map (choose Preroute > Power Network Voltage Drop Map) of the power network, which consists of the permanent and virtual power mesh and the synthesized power switch cells. An example is shown in [Figure 7-11 on page 7-54](#).

Figure 7-11 Synthesized Power Switch in IR Drop Map



6. Physically insert the power switch cells.

Once you are satisfied with the IR drop results and the location of the switch cells, you need to physically insert the synthesized power switch cells in the voltage area. The power switch cells are inserted in an array pattern that is uniformly distributed in the voltage area. There are two ways to do this:

- You can insert the power switch cells in the voltage area based on these coordinates, which are contained in an ASCII file that resides in the `pna_output` directory.
- You can run a tcl script file that calls the `create_power_switch_array` command to insert the power switch cells in the voltage area.

Specifying Different Multithreshold-CMOS Placement Styles

You place multithreshold-CMOS switch cells in different styles in the power-down voltage area, such as an array pattern, column pattern, or ring pattern. Power network synthesis can automatically detect the multithreshold-CMOS placement style and generate permanent and virtual power networks accordingly.

Placing Multithreshold-CMOS Cells in an Array Pattern Inside the Voltage Area

You can choose to distribute multithreshold-CMOS switch cells in an array pattern inside the voltage area. In this style, the switch cells are distributed throughout the power-down voltage area. In an array style, a permanent power supply is available across the power-down areas.

Therefore, special cells, such as retention registers and always-on buffers that require the permanent power supply, can be connected to the permanent power network in the power-down voltage areas.

The primary power pin of each switch cell is aligned with the standard cell rails of the power-down voltage areas. Power network synthesis automatically creates virtual rails to deliver power from the multithreshold-CMOS switch cells to the power-down logic inside the voltage areas.

The secondary power pin of the switch cell is connected to the permanent power straps on top of the power-down voltage area. Power network synthesis automatically aligns the permanent power straps across the voltage area with the secondary pins of the multithreshold-CMOS switch cells and connects them.

[Figure 7-12](#) shows the multithreshold-CMOS switch cells distributed in an cell array pattern inside the power-down voltage area with a defined power and ground strap.

[Figure 7-13](#) shows the IR drop map after power network synthesis is run.

Figure 7-12 Multithreshold-CMOS Switch Cells Placed in an Array Pattern Inside the Voltage Area

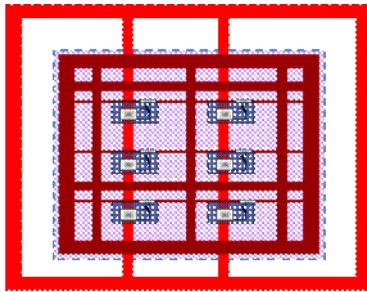
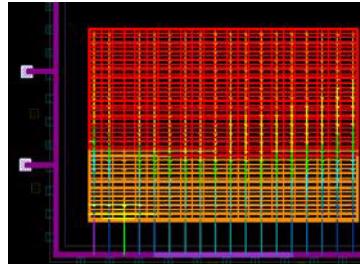


Figure 7-13 IR Drop Map After Power Network Synthesis is Run



Note:

If the permanent power net is the same as the supply power net in DEFAULT_VA, the permanent straps in the opposite routing direction of the alignment straps will crossover the power-down voltage area from the DEFAULT_VA to balance the IR voltage drop.

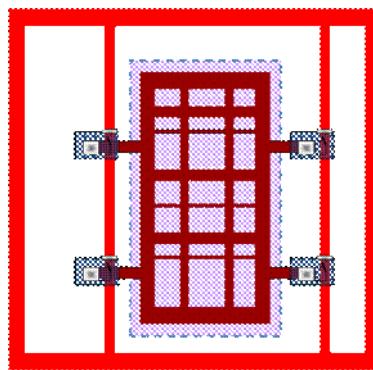
Placing Multithreshold-CMOS Cells in Columns Outside the Voltage Area

You can choose to distribute the multithreshold-CMOS switch cells in columns outside of the power-down voltage area. In a column-based placement style, the switch cells are distributed evenly across the power-down voltage area. The multithreshold-CMOS switch cells switch the power rail to each segment of the standard cell row and provide very fine control over the switch function. Because each power switch only has to provide power to a small segment of the standard cell row, potential voltage drop problems are minimized. The disadvantage of a column-based style is the distribution of the switch cells in a column can impact the placement optimization, such that the columns of switch cells act as regularly spaced placement blockages, limiting the flexibility of the placement engine.

There are two different ways to place power switch cells in a column style.

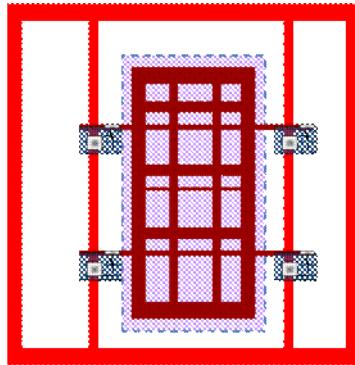
[Figure 7-14](#) shows the primary power pin of a multithreshold-CMOS switch cell aligned with the standard cell rail outside the power-down voltage area. Power network synthesis delivers power from the switch cells to the logic cells inside the power-down voltage area by generating additional wire segments that connect the secondary power pin of the switch cells to the virtual power network. To improve the IR drop results, power network synthesis also aligns the permanent power straps with the primary power pin of the multithreshold-CMOS switch cells and connects them.

Figure 7-14 Permanent Power Pin of the MTCMOS Switch Cell Aligned With the Standard Cell Rail Outside the Power-Down Voltage Area



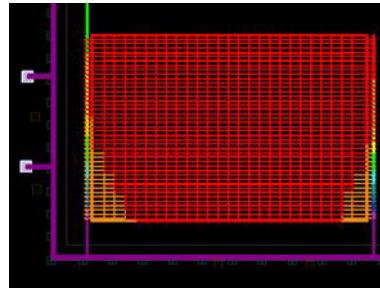
[Figure 7-15 on page 7-57](#) shows the primary power pin of a multithreshold-CMOS switch cell aligned with the standard cell rail inside the power-down voltage area. Power network synthesis delivers power from the switch cells to the power-down logic cells by extending the standard cell rail of the virtual power net from the power-down voltage area to the primary power pins of the switch cells. Power network synthesis aligns the permanent power straps with the secondary power pin of the switch cells.

Figure 7-15 Primary Power Pin of the MTCMOS Switch Cell Aligned With the Standard Cell Rail Inside the Power-Down Voltage Area



[Figure 7-16](#) shows the IR drop map after power network synthesis is run.

Figure 7-16 IR Drop Map Result After Power Network Synthesis is Run



Note:

A row-based placement style might be a more optimal for distributed switching since the potential impact on the placement engine is limited as all the switch cells are placed in a single row. This style takes a row of standard cells away from the placer, but it should not impact the placement of logic in other rows of the design. The disadvantage is that it can impact routing resources in the lower metal layers. This is avoided in the column-based placement style because lower layer power straps can be routed in metal2 directly above the power switches with minimal impact to the routing resources.

Placing Multithreshold-CMOS Cells in a Ring Pattern Around the Voltage Area

When you place multithreshold-CMOS switch cells in a ring pattern around the power-down voltage area, each switch cell is treated as a hard macro.

The ring style should be considered when area is not the main concern, and there is no need for a permanent power supply in the power-down voltage areas. A ring style has the following advantages:

- It has a less complex power plan than the array placement style because the permanent power network and the virtual power network are separate. The multithreshold-CMOS switch cells are confined in regions around the virtual power network and are not mixed with other logic cells.
- It does not have much impact on placement and routing in the standard cell area because neither the permanent nets nor the switch cells are in areas where logic cells are placed and signals are routed. Special cells such as isolation cells and always-on buffers that require the permanent power supply can be placed around the power-domain areas.

The ring style is also a good choice for small blocks of logic where the IR drop across the switch cells and the internal power network can be managed.

Power network synthesis creates two rings, a permanent power ring and a virtual power ring, and aligns them with the corresponding pins on the multithreshold-CMOS switch cells. The permanent power ring is connected to the permanent power nets outside the power-down voltage area, and the virtual power ring is connected to the virtual power nets inside the power-down voltage area.

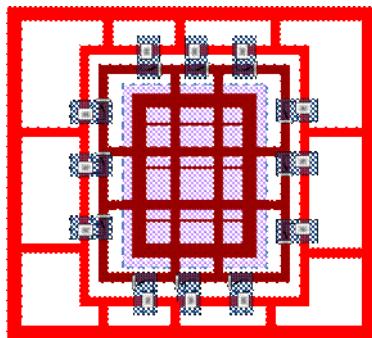
Note:

In addition to the virtual power ring aligned with the multithreshold-CMOS switch cells, by default a voltage area ring of the virtual power net is also created inside the power-down voltage area. You can disable the creation of this internal voltage area ring by using the following option:

```
set_fp_rail_voltage_area_constraints -voltage_area voltage_area_name  
-skip_ring
```

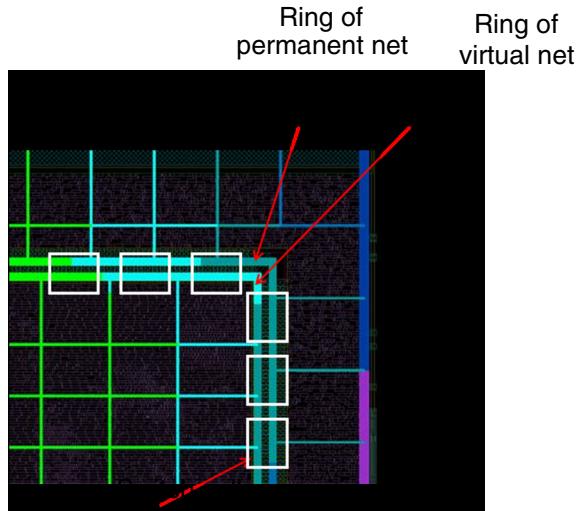
[Figure 7-17](#) shows the multithreshold-CMOS switch cells placed in a ring pattern around the voltage area.

Figure 7-17 MTCMOS Switch Cells Placed In a Ring Around the Voltage Area



[Figure 7-18 on page 7-59](#) shows the IR drop map after power network synthesis is run.

Figure 7-18 IR Drop Map After Power Network Synthesis is Run



Simulating the IR Drop Across the Multithreshold-CMOS switch Cells

Power network synthesis will simulate the static IR drop across the multithreshold-CMOS switch cells. The switch cells are modeled as a pseudo vias with resistance equal to multithreshold-CMOS switch cell on-resistance.

Before you run power network synthesis, specify the attributes of the multithreshold-CMOS switch cell on-resistance, and the name and layer of the primary and secondary power pins.

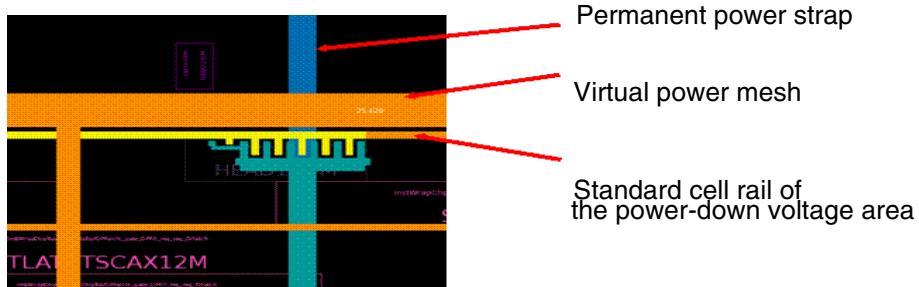
The syntax is

```
set_attribute [get_physical_lib_cells switching_cell_name]
"mtcmos_resistance" resistance_value

set_attribute [get_physical_lib_cells switching_cell_name]
"mtcmos_pin_layers" "pin1_name pin1_layer pin2_name pin2_layer"
```

[Figure 7-19 on page 7-60](#) shows the IR drop across the multithreshold-CMOS switch cells.

Figure 7-19 Simulating the IR Drop Across MTCMOS Switch Cells

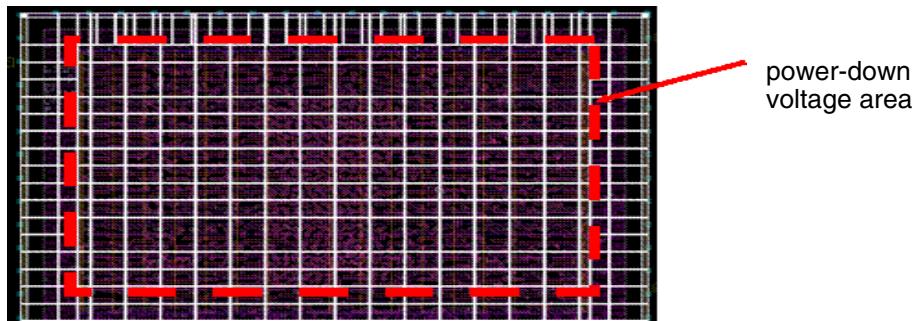


Synthesizing the Common Power Network

If the permanent power net of the power-down voltage area is also the power supply net in the DEFAULT_VA, power network synthesis will connect the permanent power net between the power-down voltage area and the DEFAULT_VA, and simulate the IR drop in the common power network.

[Figure 7-20](#) shows a common power network.

Figure 7-20 Common Power Network



Adding the Power Budget and Power Straps

While synthesizing the power net in DEFAULT_VA, power network synthesis adds the power budget in the power-down voltage area and DEFAULT_VA together, and uses it as the total power budget for IR drop simulation in DEFAULT_VA. To improve the IR drop of the permanent power net, power network synthesis also adds power straps in DEFAULT_VA to route across the power-down voltage area that uses the same power net as the permanent power supply.

Advanced Strategy to Control Aligned Permanent Power Straps

By default, power network synthesis aligns the primary power straps with the corresponding multithreshold-CMOS switching cell pins to connect them. The width of the aligned primary power straps is the width of the pin shape, and the layer is the topmost metal layer in the same routing direction.

For the array style, the primary power straps are aligned vertically with the multithreshold-CMOS switching cells, and for the column style and ring style, the primary power straps are aligned based on the placement of the multithreshold-CMOS switching cells.

You can modify these default configurations by using the following command and specifying the alignment information in a constraints file.

The `set_fp_rail_strategy` command uses the following syntax:

```
set_fp_rail_strategy -align_strap_with_mtcmos_cells
alignment_constraints_file_name
```

In the configuration file, which is a simple ASCII file, enter the following information, one net per line:

```
net_name direction [width] [layer] [voltage_area_name]
```

net name is the name of the permanent power net to which the multithreshold-CMOS cell is aligned.

direction is the direction of the alignment straps, either horizontal, vertical, or ring.

width is the width of the aligned primary power straps. By default, the width of other power network synthesis straps on the same layer is used.

layer is the metal layer name of the aligned primary power straps. By default, the top metal layer in the same direction specified in the power network synthesis layer constraints is used. In a ring style, *layer*, should specify both horizontal and vertical ring layer names.

voltage_area_name is the name of the voltage area to which the voltage area strategy is applied. The default is the entire design.

The following example shows how to specify a configuration file. In this example, there are two voltage areas: VA1 and VA2. Both voltage areas are power-down voltage areas. VA1 uses permanent power net VDD1. VA2 uses permanent power net VDD2. VA1 uses an array style for multithreshold-CMOS placement and VA2 uses a ring style for multithreshold-CMOS placement.

The contents of the `example.align` file are:

```
VDD1 vertical 1.5 METAL6          VA1
VDD2 ring      1.0 METAL5, METAL6 VA2
```

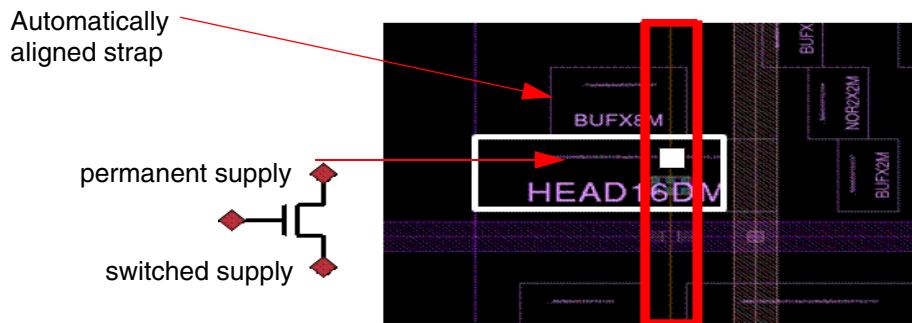
Note:

If you want power network synthesis to use the default values for the *width* and *layer* fields, use the keyword *default*, as follows:

```
VDD1 vertical default METAL6 VA1
```

[Figure 7-21](#) shows strap alignment with pins of multithreshold-CMOS switch cells.

Figure 7-21 Strap Alignment With Multithreshold-CMOS Switch Cells



Defining Multivoltage Power Network Synthesis Constraints in a Specified Voltage Area

You can set different multivoltage power network synthesis constraints in a specified voltage area.

Choose Preroute > Multivoltage Power Network Constraints > Voltage Area Constraints. The Voltage Area Power Network Constraints dialog box appears.

Alternatively, you can use the `set_fp_rail_voltage_area_constraints` command.

You can use this dialog box to interactively control and view the power network synthesis constraints assigned for voltage areas.

The constraints are grouped into the following four categories on the voltage area power network constraints dialog box.

- Synthesis constraints (default)

You can specify the power and ground net names, supply voltage, power budget, and target voltage drop in the voltage area.

- Power Ground nets – Enter the power and ground net names to be synthesized in the voltage area. If the voltage area is a power-down voltage area, specify the permanent and virtual power and ground net as “*permanent_net + virtual_net*”.

If the Unified Power Format (UPF) commands define the power-down voltage area, power network synthesis automatically retrieves the permanent and virtual net information from the UPF version. You do not need to specify the “Power Ground nets” option.

- Supply Voltage (V) – Enter the voltage supply in the voltage area. The units are measured in volts.
- Power budget (mW) – Select this option and specify a power budget in the voltage area. The units are measured in milliwatts. By default, power budgets are automatically assigned to each voltage area based on the total budget specified in the Synthesize Power Network dialog box.
- Target IR drop – Specify a target IR drop value in the voltage area. The units are measured in millivolts. The default is 10 percent of the supply voltage in that voltage area.
- Lib cell for power switch – Select this option and enter the name of the library cell to be used as the power switch in the voltage area. Use the cell reference library name in non-UPF mode, and use the power switch cell name defined by UPF in UPF mode. Use this option only in power switch synthesis.

- Global constraints

Select this option to specify the global power network synthesis constraints in the specified voltage area.

- Remove floating segments – Select this option if you want power network synthesis to remove dangling wire segments cut by hard macros or blockages that are inside the voltage area. By default, dangling wires are removed from the voltage area.
- Force same width sizing – Select this option if you want power network synthesis to create power and ground straps with the same widths in the voltage area. By default, power network synthesis creates straps with different widths for power and ground nets in the voltage area.
- Generate stacked vias – Select this option if you want stacked vias in the voltage area (the vias are dropped only on adjacent layers). By default, power network synthesis generates stacked vias.
- Optimize tracks usage – Select this option if you want power network synthesis to optimize (size) the width of the wires in the voltage area so that enough space is left for the power and ground wire’s adjacent tracks to be used for signal routes. The default is off.

- Ignore all the blockages – Select this option if you want power network synthesis to generate power and ground straps in the voltage area, despite the presence of hard macro blockages. The default is off.
 - No routing over plan groups – Select this option if you do not want power network synthesis to route the power and ground straps across plan groups in the voltage area, despite the absence of blockages. The default is off.
 - No routing over hard macros – Select this option if you do not want power network synthesis to route the power and ground straps across hard macros in the voltage area, despite the absence of blockages. The default is off.
 - No routing over soft macros – Select this option if you do not want power network synthesis to route the power and ground straps across soft macros in the voltage area, despite the absence of blockages. The default is off.
 - Allow straps over other voltage areas – If you enable this option, power network synthesis can synthesize the power network in nested voltage areas while allowing the routing of regular voltage area power and ground straps to route across other voltage areas that use different power supply nets. This allows a voltage area nested inside another voltage area to access the direct power supply from the power pads or core rings outside the nested voltage areas when running multivoltage power network synthesis.
- Layer constraints
- Select this option to specify the power strap configuration on each metal layer in the specified voltage area.
- Layer – Specify the name of the power and ground strap metal layer in the voltage area.
 - Direction – Specify the direction (vertical or horizontal) of the metal layer power and ground straps in the voltage area. The default is vertical.
 - Density – Enter the maximum number of power and ground straps in the voltage area. The default is 128.
Enter the minimum number of power and ground straps in the voltage area. The default is 16.
 - Enter the maximum pitch and minimum pitch of the power and ground straps in the voltage area.
 - Width – Enter the maximum width and minimum width of the power and ground straps in the voltage area. By default, power network synthesis uses the maximum and minimum width defined in the technology file.

- PG spacing – Specify the spacing between power and ground straps in the voltage area. You can use minimum spacing (the default), interleaving as a keyword, or you can explicitly enter a floating point number in microns.
- Offset – Enter the distance from the voltage boundary to the left-most and bottom-most power straps. The units are measured in microns. By default, power network synthesis calculates the offset distance based on the number, width, and pitch of the straps.
- Ring and strap constraints

Select this option to specify the voltage area ring constraints in the voltage area. For the DEFAULT_VA, this ring constraint is used to generate the core ring in the design. You can also specify the number of straps extended out to the core ring in the voltage area.

- Generate Rings – Specify the net names of the power and ground rings to be generated inside the voltage area. By default, the names of the ring nets are the same as the names of the synthesis nets in the voltage area.

To generate the core ring for each voltage area, you must specify the ring nets in the DEFAULT_VA. The name of the voltage area ring is the same as the core ring. You can specify multiple power ring net names, one for each voltage area, in the DEFAULT_VA.

- Horizontal layers – Specify the horizontal metal layer for power and ground rings in the voltage area. By default, power network synthesis uses the highest horizontal metal layer.
- Vertical layers – Specify the vertical metal layer for power and ground rings in the voltage area. By default, power network synthesis uses the highest vertical metal layer.
- Ring width – Specify the width of the power and ground ring in the voltage area. The units are measured in microns. By default, the ring width is twice the strap width in the voltage area, except for DEFAULT_VA. In DEFAULT_VA, the default ring width is 10 times the minimum width defined in the technology file for the specified ring metal layer.
- Ring spacing – Specify the distance between the power and ground rings in the voltage area. (Optional) The units are measured in microns. By default, power network synthesis uses the minimum spacing from the technology file.
- Ring offset to IO or PNS region – Specify the offset from the I/O pads to the power rings or the distance from the voltage area boundary to the closest ring net. The units are measured in microns. By default, the ring offset is 0.
- Extend straps to – If you choose to generate a power plan without a core ring, you need to instruct power network synthesis on how to connect (extend) the generated horizontal and vertical power straps. There are three ways to do this:

Core ring – When this option is enabled (the default), power network synthesis extends the power straps to the existing core ring in DEFAULT_VA.

Note that not all the straps in the voltage area are extended out to the core ring. The number of straps extended from each side of the voltage area is controlled by the “Direction for extending straps” option.

Top boundary – When this option is enabled, power network synthesis extends the power straps to the top-level cell boundary and creates power pins.

Pad ring – When this option is enabled, power network synthesis extends power straps to an existing pad ring. When this option is disabled, the straps stop at the pad ring boundary.

- Direction for extending straps – When this option is enabled, power network synthesis extends the direction of the straps (top, bottom, left, right, or a combination) from the voltage area to the core ring or the boundary. By default, the straps are extended from all sides of the voltage area.

Removing Voltage Area Constraints

You can remove the multivoltage power network synthesis constraints from a specified voltage area or from all voltage areas by using the `remove_fp_rail_voltage_area_constraints` command.

The `remove_fp_rail_voltage_area_constraints` command uses the following syntax:

```
remove_fp_rail_voltage_area_constraints
-all
-voltage_area voltage_area_name
-layer layer_name
```

Alternatively, you can use the Remove All button in the Voltage Area Power Network Constraints dialog box.

Reporting Voltage Area Constraints

You can generate a report of the power network synthesis constraints for power planning, based on user-defined or default settings, by using the `report_fp_rail_voltage_area_constraints` command. It generates reports for layer constraints, core ring constraints, block ring constraints, global constraints, and multivoltage power network synthesis constraints for voltage areas.

The `report_fp_rail_voltage_area_constraints` command uses the following syntax:

```
report_fp_rail_voltage_area_constraints
-all
```

```
-voltage_area voltage_area_name
-layer layer
-ring
-global
```

Example:

```
icc_shell> report_fp_rail_voltage_area_constraints -voltage_area INS
    report voltage area 'INST' constraints
Voltage area name: INST
Net name:          VDD1,VSS
Voltage supply:    1.10 [V]
Power:             122.00 [mW]
```

Allocating Power Budgets to Each Voltage Area

After you specify the constraints for each voltage area, multivoltage power network synthesis performs power budget allocation for the specified voltage areas based on the supply voltage value that you specify (`-voltage_supply` option of the `set_fp_rail_voltage_area_constraints` command) and the power budget in the voltage area (`-power_budget` option of the `set_fp_rail_voltage_area_constraints` command). The power unit is in milliwatts. The default is 1000.

By default, power budgets are automatically assigned to each voltage area based on the total power budget specified in the `synthesize_fp_rail -power_budget` command. Power network synthesis distributes the total power to all placed instances according to their area ratio.

Synthesizing the Voltage Areas

You can synthesize power nets and a common ground net on multiple voltage areas within a single power network synthesis run based on the power plan constraints. Multivoltage power network synthesis identifies all voltage areas as either DEFAULT_VA in the top-level design or non-DEFAULT_VA (sub-voltage areas). A power budget has been assigned to each voltage area.

You can synthesize all voltage areas or a specified voltage area or areas.

To synthesize all voltage areas, enter the following command:

```
icc_shell> synthesize_fp_rail -synthesize_voltage_areas
```

To synthesize a specified voltage area or areas, enter the following command:

```
icc_shell> synthesize_fp_rail -synthesize_voltage_areas -voltage_areas
voltage_areas
```

Multivoltage power network synthesis performs the following steps to support the multivoltage design flow.

1. Repeatedly synthesizes the power and ground nets on voltage areas identified as non-DEFAULT_VA.
2. Synthesizes multiple power nets and a common ground net on voltage areas identified as DEFAULT_VA. Multivoltage power network synthesis connects the common ground nets among the voltage areas.

Generating Multi-Net Core Rings and Connecting Power Straps

In multivoltage power network synthesis, each voltage area can have different supply voltages. The power straps in those voltage areas must connect to their power sources at the top level. To do this, a core ring is created for each power net at the top level, that is, in the default voltage area (DEFAULT_VA). The core rings are then connected to the power straps or voltage rings in each voltage area.

More specifically, you specify the names of different multiple ring nets in the DEFAULT_VA (`set_fp_rail_voltage_area_constraints -ring_nets nets`) and create them inside each voltage area. Multivoltage power network synthesis first generates multiple core ring nets in the DEFAULT_VA. It then connects the power nets in each voltage area to their corresponding core rings while synthesizing the power network inside each voltage area.

To reduce the number of straps extended out to the DEFAULT_VA, which might potentially cause DRC violations with the power nets in the DEFAULT_VA, you can use the `-num_extend_straps` constraint to specify the number of power straps extended out from each side of the voltage area to the core ring. The default is 4.

Note:

The ground net is not extended out from each voltage area to the core ring because the ground net is synthesized as a common net in the design. As a result, the ground net in each voltage area will be extended to the nearest ground straps outside the voltage area boundary.

```
set_fp_rail_voltage_area_constraints \
-voltage_area DEFAULT_VA \
-ring_nets {VDD, VDD1, VDD2} \
-ring_width 2.0

set_fp_rail_voltage_area_constraints \
-voltage_area INST \
-nets VDD1 \
-voltage_supply 1.2 \
-power_budget 240 \
-num_extend_straps 8
```

Supporting Incomplete Power Network Analysis

To support incomplete power network analysis, multivoltage power network analysis uses virtual rails to connect multithreshold-CMOS cells with other cells in each non-DEFAULT_VA, thereby removing the dependency on preroute commands.

Committing the Synthesized Results

Once the IR drop map meets the IR drop constraints, run the `commit_fp_rail` command to commit all the synthesized power and ground nets simultaneously. The command transforms the IR drop map into a power plan and generates a real power plan with net wires and vias.

Performing Power Network Synthesis

As the design process moves toward smaller geometries, issues related to power and signal integrity, such as power grid generation, voltage (IR) drop, and electromigration, have become more significant and complex. In addition, this complex technology lengthens the turnaround time needed to identify and fix power and signal integrity problems.

Before committing to a detailed power plan structure, you can perform power network synthesis to get a rough power plan estimation by experimenting with different user-defined power topologies to create power and ground meshes with rectangular or rectilinear power rings.

By performing power network synthesis, you can preview an early power plan that reduces the chances of encountering electromigration and voltage drop problems later in the detailed power routing.

This section includes the following topics:

- [Why Use Power Network Synthesis?](#)
- [Opening the Power Network Synthesis User Interface](#)
- [Performing Power Network Stacked Via Removal to Reduce Congestion](#)
- [Specifying Global Constraints for Power Planning](#)
- [Specifying Core Ring and Strap Constraints for Power Planning](#)
- [Specifying Layer Constraints for Power Planning](#)
- [Specifying Constraints for Block Ring Synthesis](#)

- Specifying Constraints for Power Pad Synthesis
- Specifying Regions for Power Network Synthesis
- Performing Region-Based Nonuniform Mesh Density Power Network Synthesis
- Creating User-Defined Power and Ground Routing Constraints in Hard and Soft Macros
- Defining Customized Strap Configurations for Hard Macros and Rectangular Regions
- Aligning Power Network Synthesis Straps With Top-Level Pins
- Resynthesizing a Power and Ground Mesh Inside a Soft Macro
- Aligning Power and Ground Straps With Standard Cell Rails
- Placing Power and Ground Straps Inside Standard Cell Rows
- Performing Strap Alignment With Area I/O Cells
- Committing the Power Plan
- Checking the Integrity of the Power Network
- Handling TLUPlus Models in Power Network Synthesis
- Specifying the Operating Temperature for IR Drop Analysis in Power Network Synthesis and Power Network Analysis
- Supporting the Unified Power Format (UPF) Mode in Power Network Synthesis

Why Use Power Network Synthesis?

Power network synthesis offers advanced power planning technology and helps solve signal integrity problems without lengthy and tedious iterations. By performing power network synthesis, you can preview an early power plan and thereby reduce the chance of encountering electromigration and voltage drop problems later in the detailed power routing. Power network analysis also helps you avoid overdesigning the power plan early in the design cycle because you know the electromigration and voltage drop effects in the power planning stage.

Power network synthesis provides these benefits:

- Produces automated IR-aware power planning

Power network synthesis allows you to enter the IR drop as a constraint, along with trunk width and mesh density constraints, so that you can quickly create a power mesh in multimillion-instance designs.

- Manages an IR drop strategy before power plan creation

The preview feature in power network synthesis gives you early insight into IR drop's impact on the power structure before a single power wire is created. If your virtual power mesh, generated by power network synthesis, does not meet with your satisfaction, you can revise the constraints in power network synthesis and then regenerate the power plan. Because power planning creation in power network synthesis takes only few minutes (even on a multimillion-instance design), you can easily experiment with different power mesh topologies.

- Develops a production-quality power plan from preroute technology

Power network synthesis is built on top of preroute technology. It creates preroute commands and internally executes them when you use the `commit_fp_rail` command to commit the power plan. The power plan meets your specified via generation rules and fat wire rules in addition to being DRC-clean.

- Saves routing resources

The power network synthesis optimization engine helps you find the optimal power routing structure to meet IR drop specifications. Power network synthesis generates a power mesh that meets IR drop, and at the same time, it minimizes the power routing resources. When power network synthesis finishes power mesh generation, it reports the routing resources used. This information can be valuable for congestion management later in the design process.

- Drives other power planning tools

Power planning is a very customized process because your design style and power mesh topology vary from one process to another. Power network synthesis helps you generate a *starting point* power plan for reference. You can use the power plan generated by the power network synthesis as the final power plan and continue the implementation, or you can use the information provided by power network synthesis to drive customized or third-party power planning tools. An example of this might be a 25 x 25 power mesh with a 8.4-micron trunk width and a 10 percent metal5 and 12.5 percent metal6 routing resource to meet a 100 millivolt IR drop.

- Manages congestion and utilization prior to floorplanning

You can use power network synthesis to generate a power plan before instances are placed but after the die is initialized. Enter the power budget for the entire design, and use power network synthesis to estimate the power structure needed to meet the IR drop requirements. Then use the routing resources information to budget routing for signal nets. If the power structure uses too much routing resources, you can reinitialize the design with a smaller utilization ratio to meet signal routing resource needs.

- Generates a custom power mesh

You can use power network synthesis to create a power mesh (assuming that you do not want to specify IR drop constraints). To do this, set the maximum and minimum number of constraints and the width to the same value. When this runs, power network synthesis bypasses the optimization engine and generates a power mesh you have specified.

- Produces a rectilinear power mesh

One of the biggest challenges in power planning is rectilinear mesh construction. Power network synthesis provides you with one-step rectilinear power mesh generation that meets IR drop requirements, is DRC-clean, and is perfectly snapped to the power plan grid.

- Reduces the need for complex power plan editing

Power planning can sometimes be labor-intensive, because it involves more than routing a wire from one point to the other. If your power planning tools do not generate output with the concept of a power structure, you might be required to spend more time editing. Power network synthesis creates output with the concept of a power structure and avoids physical restrictions when creating the power structure. For example, rather than stopping a power trunk at a blockage, power network synthesis cuts the trunk to accommodate the blockage and snaps the line (optional) to the closest intersection. Also, you can specify that dangling wire segments be removed, which reduces the need for power plan editing.

- Prepares a power grid prototype

Because power network synthesis makes power grid creation less complex, you can use it as a power grid prototyping tool. Used in this way, you can experiment with different power mesh designs and optimize the power mesh to meet your specific design needs.

- Replays power plan creation

While power network synthesis creates the power plan, it simultaneously generates a command script containing the `create_power_straps` command that you can view and edit. If there are minor incremental floorplan changes, you can also use this script to revise the power plan. Customizing the script for your particular power planning requirements is also possible with this process.

- Detects placement problems

Power network synthesis lets you quickly create a power plan and simultaneously lets you see any placement problems that would block power trunk traffic. (For example, a hard macro with blockage on metal5 is placed in the extending power routing area.) By using power network synthesis power planning, you can budget and make tradeoff decisions based on power routing resources and optimal placement locations. After you make the placement adjustment, you can replay the power plan creation script generated by power network synthesis or simply rerun power network synthesis.

- Creates region-based power mesh generation for multivoltage support

You can use power network synthesis to easily create incremental regional power mesh structures by specifying the mesh regions one at a time.

Opening the Power Network Synthesis User Interface

The preferred time to run power network synthesis is right after virtual flat placement. At that point, the cells are closer to the final placement, so power network synthesis can do a better job of estimating power distribution. You can also run power network synthesis before the placement is finalized and after the core is initialized.

To open power network synthesis,

1. Choose Preroute > Synthesize Power Network.

The Synthesize Power Network dialog box appears.

Alternatively, you can use the `synthesize_fp_rail` command.

This is where you enter all the information for your power plan. You can get a rough power plan estimation by experimenting with different user-defined power topologies to create rectangular or rectilinear power rings before committing to a detailed power plan structure.

You can synthesize a power network for a single voltage design, for a multivoltage design (see “[Performing Power Network Synthesis On Multivoltage Designs](#)” on page 7-50), or for a power switch array based on user-specified constraints.

2. Set the options, depending on your requirements.

- In the “Synthesis power network by net” field, specify the names of the power and ground nets on which to perform the power network analysis. Both nets are processed simultaneously.
- Perform constraint-driven power network synthesis by specifying a total power budget for the analyzed design.

The power budget is divided among the instances according to their size. The power unit is milliwatts. The default is 1,000.

For hard macros or standard cells, the power budget is computed based on the percentage of the total area. For a hierarchical block, the power budget is computed based on the sum of all the cells and blocks inside it. If there are no logical connections to the analyzed net, the allocation is zero.

For example, if the power budget for the entire design is 500 milliwatts, the total area of the top-level blocks is 500,000 square microns, and one of the blocks has an area of 25,000 square microns, power network analysis allocates the power for that 25,000-square-micron block as follows:

$$500\text{mW} * (25,000/50,000) = 25\text{mW}$$

Note:

You can also obtain the power calculation source from a PWR view (for hard macros) or from Power Compiler and PrimePower report files.

- Specify a voltage supply by entering the voltage of the net you want synthesized. The unit is volts. The default is 1.5.
- Specify the target IR drop value. The default is 10 percent of the power pad voltage supply. If the voltage supply is 1.5 volts, the default drop would be 150 millivolts.

Select the Lowest option if you want to direct the synthesize engine to synthesize a power plan or a pad that would give the lowest-possible IR drop value based on user-defined constraints.

- Specify the “synthesize power plan” option to perform normal voltage power network synthesis. This is the default behavior. You can specify this option together with the “synthesize power pads” option to perform simultaneous power network and power pad synthesis.
- Specify the “synthesize power network by voltage area” option to perform multivoltage power network synthesis on voltage areas that have been specified by using the `set_fp_rail -voltage_area constraints` command. (See [“Performing Power Network Synthesis On Multivoltage Designs” on page 7-50](#).)

Select the “All voltage areas” option to synthesize power networks in all voltage areas that have been specified by using the `set_fp_rail -voltage_area constraints` command. If you want to specify a voltage areas in which to perform multivoltage power network analysis or power switch synthesis, select the “Specified voltage areas” option, and enter the name of the voltage area.

- Specify the “synthesize power switch array” option to perform power switch synthesis on the specified powered-down voltage area
- Specify the power and ground pad information. Pad information is critical for power network synthesis. By default, power network synthesis assumes that any power pad with a logical connection to the power net is the power source for the power net. However, this assumption might not hold where pads have a logical power net connection that is used to power the pad signal driver, such as a power pad ring. In such a case you might want to specify an exact power pad instance or master.

The following four power and ground pad definition types are supported:

Type 1: Specifying the input design pad file name – Enter the name of the pad file.

If the Instances option is active, you can enter a power and ground pad definition file in instance-based format.

The format is

instance_name net_name

(The *net_name* is optional.) For example,

VDD1 VDD

If the Masters option is active, you can enter a power and ground pad definition file in master-based format. This is the default.

The format is

master_name net_name

(The *net_name* is optional.) For example,

VDD.FRAM VDD

Type 2: Specifying pad masters – Enter the name of the pad master in the text box.

Type 3: Use top-level pins as pads – Select this option if you want pins to be regarded as pads during block-level simulation.

Type 4: Use strap ends as pads – Use this option during block-level power planning when top-level power and ground pins are missing. Power network synthesis will use the ends of straps at the design boundary or core ring as power and ground pads. Because the pads are defined, you do not need to define virtual pads and additional pad information.

By default, power network synthesis places the power and ground pads at four sides of the strap end: north, south, east, and west. If you want to control which direction the sides of the strap ends are used as power and ground pads, set the following environment variable:

PNS_BOUNDARY_PAD_SIDE

Power network synthesis creates a “straps_end. *netname.vpad*” virtual pad file to store the power pad information defined by this option.

- Specify a hierarchical option.

If your design contains macro cells that are placed and routed, you can sequentially search downward into these macro cells and bring all of the power and ground segments up to the top-level resistance network.

Flatten Hierarchical Cells (the default) – The power network synthesis engine analyzes the design as if it were flat.

Top Level Cells Only – The power network synthesis engine ignores cells inside soft macros but considers all the power and ground net wires (and vias) in the design.

- Specify the extraction options.

Create virtual rails – During the floorplanning stages, metal1 straps for the standard cell pin connections are usually not available. Power network synthesis uses this option to create virtual pseudo straps. By default, metal1 is used to generate the

pseudo straps for the standard cell pin connections. Power network synthesis uses the virtual straps for analyze the IR drop and electromigration to predict what the effects might be when the metal1 straps are available.

Horizontal pseudostraps are created with specified layers for standard cell pin connections, based on the row information in the database.

Ignore CONN view layers – Reduces memory usage in power network analysis and power network synthesis and speeds up the IR simulation.

If you select the All option, all the metal layers in the CONN (connectivity) views are ignored.

If you select the Specified option, click the mask names of the layers to be ignored in the CONN views.

- Specify power information – There are four ways to obtain the power calculation source required to analyze the power:

Power consumption file – To manually set power on a cell instance or cell master through a file, enter the name of the power consumption file. This file specifies the amount of power consumed by each instance. Select the file format: Default/AstroRail, Power Compiler, or Prime Power/PrimeTime-PX.

Default/AstroRail format – To manually specify a power budget, select the Default/AstroRail option. In the data file, use the following format to obtain the power calculation of each instance, in milliwatts:

<instance_or_master_name> name power_value (mW)

For example:

```
instance X1 0.2  
master abc.FRAM0.1
```

This is the format of the file produced by the AstroRail `poDumpPowerInfo` command.

Power Compiler – To obtain the power calculation source from a Power Compiler report file, select the Power Compiler option. After you complete a power calculation in Power Compiler, use the following command to generate the report file:

```
report_power -cell -flat -hier -nosplit report_file_name
```

PrimePower/PrimeTime-PX – To obtain the power calculation source from a PrimePower or PrimeTime-PX report file, select this option. After you complete a power calculation in PrimePower, use the following command to generate the report file:

```
calculate_power -waveform
```

If `-waveform` is specified, PrimePower generates both instance peak current and average current information. Alternatively, use:

```
report_power -file report_file_name -leaf
```

The `-leaf` option is used for generating the power of all the leaf and block instances.

PrimeTime PX – After you complete a power calculation in PrimeTime PX, use the following command to generate the power file:

```
report_power -cell_power -leaf file_name
```

- Specify an output directory.

The analysis results are output to the default directory, `pna_output`. The specified directory stores the IR drop and electromigration results with the file name `design_name.net_name.pw_hi.pna` and the power allocation results with the file name `design_name.net_name.power`.

The IR drop and electromigration results are stored in the analysis file:

```
./pna_output/ designname.netname.pw_hi.pna
```

The power allocation results of the computed cells and blocks are stored in the analysis file:

```
./pna_output/ designname.netname.power
```

Note:

Because these files can be quite large, make sure you have enough disk space in your `./pna_output` directory.

3. Click OK or Apply.

Performing Power Network Stacked Via Removal to Reduce Congestion

As the voltage-level in designs continues to scale down, the noise introduced by voltage IR drop in the power or ground network becomes more critical. These designs may require a denser power network to minimize the voltage drop. However, this can cause the creation of a large number of stacked vias from the top mesh down to the standard cell rails. These stacked vias may result in congestion issues. You can use the `reduce_fp_rail_stacked_via` and `remove_fp_rail_stacked_via` commands to identify and select a set of stacked vias in a complete or partially-built power network for removal to reduce congestion and then to remove these via objects physically from the design.

Selecting Stacked Vias for Removal

Given a complete or partially-built power network, you can use the `reduce_fp_rail_stacked_via` command to identify all removable stacked vias in the power network. A set of critical wires in the least congested area of the power network is identified to ensure that each originally connected standard cell or macro block can receive power from power pads through this set of wires. All stacked vias that are in a congested area and do not contain any critical vias are identified as “feasible” stacked vias. You can select these vias for removal such that the voltage drop constraint is honored and the connectivity of the existing power network is maintained.

The `reduce_fp_rail_stacked_via` command does not remove signal stacked vias, stacked vias in noncongested areas, stacked vias connected to power or ground ring segments, stacked vias used to block macro pin connections, or stacked vias in soft macros when the top-level design is open.

Before the selected stacked vias objects are physically removed from the design, you can preview the results of removing the stacked vias in terms of voltage drop and electromigration effects. (See [Removing the Selected Stacked Vias](#).) You can also get an estimate of the total congestion overflow reduction based on the existing congestion map if the selected stacked vias were removed.

Note:

If the congestion map result is not already available, the `route_global` command is invoked in the background to generate the map. This can result in additional memory and runtime usage.

The `reduce_fp_rail_stacked_via` command uses the following syntax:

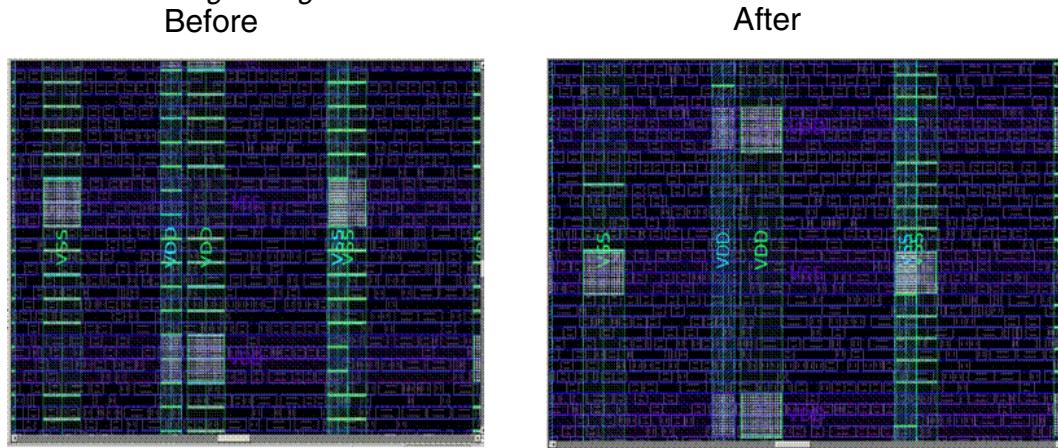
```
reduce_fp_rail_stacked_via
-nets
[-power_budget power]
[-analyze_power]
[-lowest_voltage_drop]
[-target_voltage_drop target_voltage]
[-voltage_supply voltage]
[-effort low | medium | high]
[-pad_lib_cells pad_lib_cells]
[-read_pad_instance_file file_name]
[-read_pad_lib_cell_file file_name]
[-extract_stacked_via_only]
[-use_pins_as_pads]
[-top_level_only]
[-ignore_blockages]
[-ignore_conn_view_layers layer]
[-read_power_compiler_file file_name]
[-read_prime_power_file file_name]
[-read_default_power_file file_name]
[-output_directory directory_name]
```

Removing the Selected Stacked Vias

Once you have selected a set of power network stacked vias for removal by using the `reduce_fp_rail_stacked_via` command, you can physically remove the power network stacked vias by using the `remove_fp_rail_stacked_via` command. The command purges the selected via objects from the design based on the stacked via reduction results.

[Figure 7-22](#) shows the removal of a large number of stacked vias. This can provide more routing resources for some congested areas.

Figure 7-22 Removing a Large Number of Stacked Vias



Specifying Global Constraints for Power Planning

You can specify power network synthesis global constraints for power planning.

To specify global constraints,

1. Choose Preroute > Power Network Constraints > Global Constraints.

The Global Power Network Constraints dialog box appears.

Alternatively, you can use the `set_fp_rail_voltage_area_constraints` command.

Set the options, depending on your requirements.

- Remove floating segments – If you enable this option, power network synthesis removes any dangling wire segments cut by hard macros or blockages. This reduces the need for power plan editing. Disable this option if you do not want floating segments to be removed. The default is enabled.
- Force same width sizing – If you enable this option, power network synthesis creates wires that are the same width for both power and ground nets. If you deselect this option, power network synthesis creates wires with different widths for the power and ground nets. The default is enabled.

- Keep rings outside of core area – If you enable this option, power network synthesis creates the core ring outside of the core area. The default is disabled.
- No routing over plan groups – If you enable this option, power network synthesis will not route over plan groups, despite the absence of blockages. The default is disabled.
- Generate stacked vias – If you enable this option, power network synthesis uses stacked vias. If this option is disabled, power network synthesis drops vias only on adjacent layers. The default is enabled.
- Optimize tracks usage – If you enable this option, power network synthesis sizes wires so that it leaves enough space for the power and ground wire's adjacent tracks to be used for signal routes. The default is disabled.
- No routing over hard macros – If you enable this option, power network synthesis does not route over hard macros, despite the absence of blockages. The default is disabled.
- No routing over soft macros – If you enable this option, power network synthesis does not route over soft macros, despite the absence of blockages. The default is disabled.
- Ignore all the blockages – If you enable this option, power network synthesis creates power wires despite hard macro blockages. By default, power network synthesis honors hard macro blockages in order to create a DRC-clean power plan.

2. Click OK or Apply.

Specifying Core Ring and Strap Constraints for Power Planning

You can specify power ring and strap constraints for power planning.

To specify ring and strap constraints,

1. Choose Preroute > Power Network Constraints > Ring and Strap Constraints.

The Rings and Straps Power Network Constraints dialog box appears.

Alternatively, you can use the `set_fp_rail_voltage_area_constraints` command.

Set the options, depending on your requirements.

- Generate rings

Select this option (the default is enabled) to instruct power network synthesis to generate a core power ring.

- Horizontal layer and Vertical layer

You can specify the horizontal and vertical layers for the power ring. The default horizontal layer is the highest horizontal layer in the database; the default vertical layer is the highest vertical layer in the database.

- Ring width and Ring offset

To meet the IR drop requirements, the ring width is automatically determined.

Ring width – Enter a fixed width or a range of widths for the power ring.

Ring spacing – Specify the distance between the power and ground rings in the voltage area. (Optional) The units are measured in microns.

Ring offset to IO or PNS region – Specify the offset from the I/O pads to the power rings. The units are measured in microns.

- Extend strap

If you choose to generate a power plan without a core ring, you need to instruct power network synthesis on how to connect (extend) the generated horizontal and vertical power straps. There are three ways to do this:

Core ring – When this option is enabled (the default), power network synthesis extends the power straps to the existing core ring.

Top boundary – When this option is enabled, power network synthesis extends the power straps to the top-level cell boundary and creates power pins.

Pad ring – When this option is enabled, power network synthesis extends power straps to an existing pad ring. When this option is disabled, the straps stop at the pad ring boundary.

2. Click OK or Apply.

Specifying Constraints for Generating Multiple Core Rings or Sandwich Core Rings

If a synthesized core ring is too wide, metal slotting is required. To avoid metal slotting, you can use the options in the Rings and Straps Power Network Constraints dialog box to generate multiple core rings or sandwich core rings. (The term sandwich rings refers to rings on different layers that are superimposed on each other.)

Alternatively, you can use the `set_fp_rail_voltage_area_constraints` command.

To generate multiple core rings or sandwich core rings, in the Rings and Straps Power Network Constraints dialog box,

1. In the Power Ground nets field, enter the net names multiple times to generate multiple core rings. The first net is the outermost ring.
2. Select multiple metal layers as horizontal and vertical layers to generate sandwich core rings.

For example, if you specify the power and ground nets “VDD, VSS, VDD, VSS” in the Power Ground nets box, “metal5, metal3” in the Horizontal Layers box, and “metal6, metal4” in the Vertical Layers box, power network synthesis will generate four core rings (VDD, VSS, VDD, and VSS). For each ring, each vertical segment contains two wires (metal6 and metal4), which overlap each other to create sandwich core rings. Each horizontal segment contains metal5 and metal3 wires.

3. Click OK or Apply.

Specifying Layer Constraints for Power Planning

You can specify the current layer on which to create the power grid and the direction (vertical or horizontal) for the power and ground wires.

To specify layer constraints,

1. Choose Preroute > Power Network Constraints > Layer Constraints.

The Layer Power Network Constraints dialog box appears.

Alternatively, you can use the `set_fp_rail_voltage_area_constraints` command.

Set the options, depending on your requirements.

- Layer

Specify the metal layer on which to create the power grid. The default is the highest metal layer.

- Direction

Specify the direction layer (vertical or horizontal) for the power and ground wires. The default is vertical.

- Density and Width

You can control the density by specifying the number of straps or the pitch. Enter the maximum and minimum number of straps or the pitch in the text boxes. The default is 128 and 16, respectively, for the straps.

You can control the width of the power wires by entering the maximum and minimum width in the appropriate text boxes. The default is the maximum and minimum width defined in the technology file.

Note:

If you specify the maximum and minimum number of straps or the pitch and the same maximum and minimum width, power network synthesis bypasses the optimization engine and generates a custom power mesh as specified.

- PG spacing

You can control the spacing of the power and ground wires by minimum spacing (the default), interleaving, or microns. Constraining the power and ground wire spacing might allow signal routes between the power and ground networks.

- Offset – Enter the distance from the voltage boundary to the left-most and bottom-most power straps. The units are measured in microns. By default, the offset distance is calculated based on the number, width, and pitch of the straps.
2. Click the Set button to set the constraints for the currently selected layer. Click Remove to remove the settings for the currently selected layer or Remove All to remove all of them. Click Default to return the dialog box to its default settings. When you are done setting the layer constraints, click Close.

Specifying Constraints for Block Ring Synthesis

You can automatically create rectangular or rectilinear power and ground rings around plan groups and macros when the power network straps are synthesized by power network synthesis. By adding power and ground rings, coupling capacitance is avoided between hierarchies. This method is often used to shield the most active nets, such as a clock net, for example.

To create power and ground rings,

1. Choose Preroute > Power Network Constraints > Block Ring Constraints.

The Block Rings Power Network Constraints dialog box appears.

Alternatively, you can use the `set_fp_block_ring_constraints` command.

2. Set the options, depending on your requirements.

- You can select blocks by plan group cell instances, voltage areas, or cell masters. The power and ground rings are created around the specified cell instances, plan groups, or voltage areas.
- Power Ground nets

Enter the net names for the block rings. You can enter multiple net names to create multiple block rings. The first net will be the innermost block ring. For example, to create four rings, enter

"VDD VSS VDD VSS"

- Vertical layer

Select the vertical layers in the block ring. The default is the highest layer with the preferred vertical direction. Only one layer is allowed for each block ring constraint.

Specify the width of the vertical wires. The default value is the minimum width in the technology file.

Specify the vertical offset from the block boundary to the block ring. The default value is the minimum spacing in the technology file.

- Horizontal layer

Select the horizontal layers in the block ring. The default is the highest layer with the preferred horizontal direction. Only one layer is allowed for each block ring constraint.

Specify the width of the horizontal wires. The default value is the minimum width in the technology file.

Specify the horizontal offset from the block boundary to the block ring. The default value is the minimum spacing in the technology file.

- Spacing between block ring nets

Enter a value to specify the spacing between the block rings. The default is the minimum spacing defined in the technology file.

3. Click the Set button to set the constraints for the currently selected layer. Click Remove to remove the settings for the currently selected layer or Remove All to remove all of them. Click Default to return the dialog box to its default settings. When you are done setting the block ring constraints, click Close.

Specifying Constraints for Power Pad Synthesis

You can specify constraints for power pad synthesis by using the `set_fp_power_pad_constraints` command. Power pad synthesis determines the number of power pads and their locations, based on the chip area and a target IR drop for boundary I/O designs. It can solve IR drop problems without changing the power structure of the existing design.

The `set_fp_power_pad_constraints` command uses the following syntax:

```
set_fp_power_pad_constraints
[-honor_existing_pads | -honor_even_space]
[-target_pad_current current] [-maximum_number_of_pads number]
[-save_file filename]
```

- Honor existing pads – When this option is enabled, all existing power and ground pads are honored during pad synthesis.

The default is to ignore power and ground pads during pad synthesis. However, any virtual pads that you create are always honored during pad synthesis.

- Honor even space – When this option is enabled, pads with an even amount of space between them are honored during pad synthesis. The default is disabled (off).

Note:

This option and the `-honor_existing_pads` option are mutually exclusive.

- Target pad current – Specify the target pad current value. The power pad synthesis will attempt to add power pads and place them in the most ideal locations such that the maximum pad current, voltage drop, and power pad constraints are satisfied simultaneously.
- Maximum number of pads – Enter the maximum number of pads to be considered during pad synthesis. The default is 40.
- Save file – Specify the name of the virtual pad file that is generated during pad synthesis. The default is default.vpad.

Alternatively, you can use the Power Pads Constraint dialog box. Choose Preroute > Power Network Constraints > Power Pads.

Specifying Regions for Power Network Synthesis

You can specify regions and then synthesize the power network, based on those regions.

Note:

You can create a rectilinear core ring and straps (power mesh) inside the specified region. This feature is very useful when the design has multiple VDD regions and needs multiple power meshes for each region.

To create regions,

1. Choose Preroute > Power Network Constraints > Region Constraints.

The Region Power Network Constraints dialog box appears.

Alternatively, you can use the `set_fp_rail_region_constraints` command.

2. Set a specified region to generate.

- You can specify a region by entering a list of coordinates and drawing its outline in the layout (CEL) window. Use the left mouse button to enter the region points, and click the right mouse button to close the region.

- You can specify a region by using the existing voltage area as the region outline.

Enter the voltage area name. The default value is DEFAULT_VA, which is defined as the core area excluding all other defined voltage areas.

- If you want to remove the existing region, select the “Remove region constraints” option.

3. Click OK or Apply.

Performing Region-Based Nonuniform Mesh Density Power Network Synthesis

Power network synthesis accepts predefined strap configurations to create a nonuniform power mesh for specified regions. Power network synthesis will continue to synthesize the power straps uniformly for the rest of the core area, using regular power network synthesis constraints.

Defining Predefined Straps Configurations

You can define multiple region-based power and ground strap configurations with layer, direction, width, pitch, and offset to a file. Power network synthesis generates power straps for the regions (rectangular areas) you specify.

The syntax is

```
set_fp_rail_region_constraints -load_file strap_region_file
```

The format of the straps configuration file is

```
Region xlo ylo xhi yhi
Layer mask_name Direction dir Width value Pitch value Offset value
```

For example,

```
Region 925.330 354.000 1500.000 902.000
Layer metal6 Direction Vertical Width 8.2 Pitch 60.5 Offset
6.2
Layer metal5 Direction Horizontal Width 6 Pitch 60 Offset 6
Region 125.330 154.000 300.000 300.000
Layer metal6 Direction Vertical Width 2.3 Pitch 30.5 Offset
1.2
Layer metal5 Direction Horizontal Width 3 Pitch 10 Offset 2
Layer metal5 Direction Vertical Width 1 Pitch 5 Offset 2
```

You can also define strap configurations based on the instance or master names of hard macros.

For example,

```

Instance abc
Layer metal6 Direction Vertical Width 8.2 Pitch 60.5 Offset
6.2
Layer metal5 Direction Horizontal Width 6 Pitch 60 Offset 6
Master memory
Layer metal6 Direction Vertical Width 2.3 Pitch 30.5 Offset
1.2
Layer metal5 Direction Horizontal Width 3 Pitch 10 Offset 2
Layer metal5 Direction Vertical Width 1 Pitch 5 Offset 2

```

Creating User-Defined Power and Ground Routing Constraints in Hard and Soft Macros

You can specify the power and ground routing constraints for hard or soft macros, or both, in an input file. The file defines the macro type (master or instance), the name of the macro (master name or instance name), the name of the net that will be blocked, and the name of the layer that will block the power and ground routing.

The syntax is

```
set_fp_rail_strategy -honor_macro_route_constraints file_name
```

Routing Constraints Format

The format of the routing constraints file is:

```

<#1 block_type> <#1 block_name> <#1 net_name> <#1 layer>
<#2 block_type> <#2 block_name> <#2 net_name> <#2 layer>
...

```

block_type specifies the block type: *master* | *instance*

block_name specifies the name of the block that blocks power and ground routing.

net_name specifies the name of the net being blocked.

layer specifies the layer that blocks the power and ground routing.

For example:

```

master BLENDER_1.CEL VDD METAL3
master BLENDER_2.CEL VDD {METAL3,METAL4}
instance 1_ORCA_TOP/1_BLENDER_3 VSS all

```

In this example, VDD net will not route over all instances of cell master BLENDER_1.CEL on METAL3; VDD net will not route over all instances of cell master BLENDER_2.CEL on METAL3 and METAL4; VSS net will not route over cell instance 1_ORCA_TOP/1_BLENDER_3 on all metal layers.

Note:

The `-no_routing_over_hard_macro` and `-no_routing_over_soft_macro` constraints of the `set_fp_rail_constraints` or `set_fp_rail_voltage_area_constraints` command have a higher priority than the routing constraints file. If these constraints are specified, power network synthesis will not route over any hard or soft macros on any layers.

Defining Customized Strap Configurations for Hard Macros and Rectangular Regions

You can define customized power and ground strap configurations for hard macros and particular rectangular regions. You define a region or a hard macro type, followed by the layer, direction, width, pitch, offset, and spacing.

The syntax is:

```
set_fp_rail_strategy -honor_macro_strap_config configuration_file
```

Configuration File Format

The format of the configuration file is:

```
Region xlo ylo xhi yhi
Hard macro Instance | Master name
Layer layer_name Direction Vertical | Horizontal Width value Pitch value
Offset value [Spacing | Interleaving value | True]
```

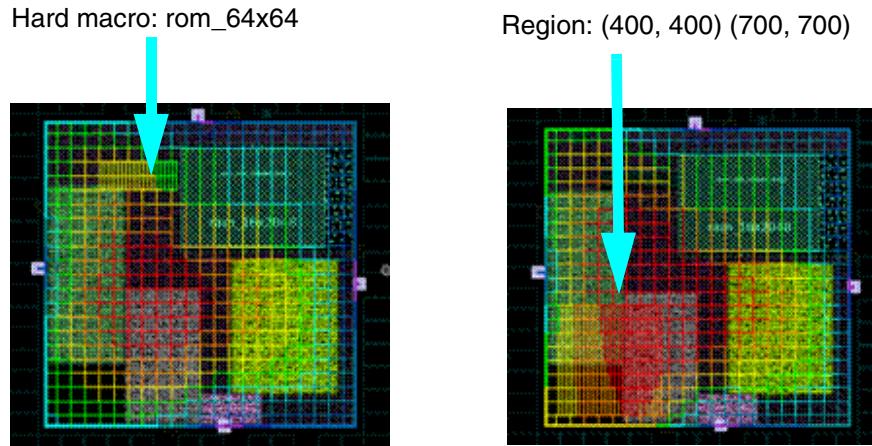
For example,

```
Master system_rom
Layer M6 Direction Vertical Width 3 Pitch 10 Offset 4 Interleaving True
Layer M5 Direction Horizontal Width 3 Pitch 10 Offset 4 Spacing 3
Layer M4 Direction Vertical Width 3 Pitch 10 Offset 4

Region 925.330 354.000 1500.000 902.000
Layer M6 Direction Vertical Width 8.2 Pitch 60.5 Offset 6.2
Layer M5 Direction Horizontal Width 6 Pitch 60 Offset 6
```

[Figure 7-23 on page 7-89](#) shows IR drop maps of strap configuration.

Figure 7-23 IR Drop Maps of Strap Configuration



Aligning Power Network Synthesis Straps With Top-Level Pins

You can specify whether or not to align power and ground straps with all the top-level power and ground pins. In hierarchical power network synthesis, you can generate a top-level power network on a design with plan groups. When the plan groups are converted into soft macros, PG pins are generated on the soft macro boundaries. If you want to later regenerate the power mesh inside the soft macros, the new power and ground straps can be aligned with the existing power and ground pins on the block boundaries. (See “[Resynthesizing a Power and Ground Mesh Inside a Soft Macro](#)” on page 7-90.)

The syntax is:

```
set_fp_rail_strategy -align_strap_with_top_pin true | false
```

If this option is set to a value of `true`, power network synthesis aligns the power and ground straps to all the power and ground pins on the block boundary that are connected to the net of the same type as the straps. This strategy is useful when performing block-level power network synthesis. The default is `false`.

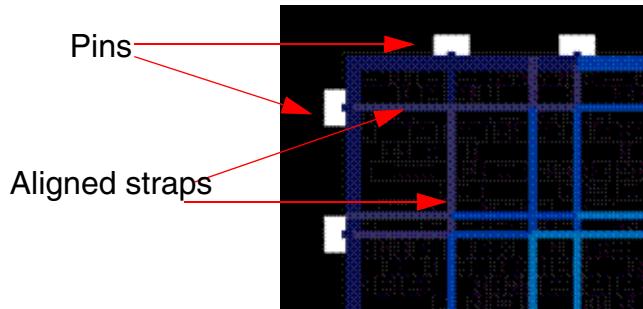
If more power and ground straps are needed to satisfy the IR drop target, power network synthesis generates additional nonaligned straps that honor the user-defined maximum or minimum strap number constraints

Note:

Aligned straps are included in the total strap number. If the maximum strap number is less than the number of pin-aligned straps, the maximum strap number constraint is not honored. When there is a DRC violation, the pin alignment straps have a higher priority than regular power network synthesis straps. However, in the case of multithreshold-CMOS alignment straps, the pin alignment straps have a lower priority.

[Figure 7-24](#) shows the width of aligned straps and layer are the same as the pins.

Figure 7-24 Strap Alignment With Top-Level Pins



Resynthesizing a Power and Ground Mesh Inside a Soft Macro

After the power network is created at the top-level of the chip in the hierarchical power network synthesis flow, the plan groups are pushed down into the soft macros. The PG mesh that resides on top of the plan groups is also pushed down into the soft macros, and the PG pins are generated on the soft macro boundaries. If you want to resynthesize the power mesh inside the soft macro, power network synthesis will generate a Tcl script file for each plan group that will be committed to a soft macro. You can run this Tcl file once during block-level power network synthesis. The PG mesh is resynthesized inside each soft macro with the same strap configuration generated on the soft macros during top-level power network synthesis and the same power budget allocated during top-level power network synthesis.

The Tcl script file uses the PG pins as the power source for the IR drop analysis in the soft macro, and it also considers the voltage drop at the PG pins.

The syntax is

```
set_fp_rail_strategy -create_hierarchical_pns_script true | false
```

By default, the strategy is set to false.

If you set this strategy to a value of true, a subdirectory `plangroup_name_PNS_script` is created under the `pna_output` directory with a name that is associated with each plan group in the directory. The following two files are generated from this subdirectory:

- The `plangroup_name.vin` file contains the voltage information for each soft macro pin on the plan group boundary.

- The `plangroup_name_replay.tcl` file is a replay file that resynthesizes the PG mesh in the soft macro.

Aligning Power and Ground Straps With Standard Cell Rails

You can align power and ground horizontal straps with power and ground standard cell rails.

The syntax is

```
set_fp_rail_strategy -align_strap_with_m1_rail true | false
```

When set to a value of `TRUE`, Power Network Synthesis aligns synthesized power and ground straps in the lowest horizontal layer with the horizontal power or ground rails of standard cells. By using this strategy, you can avoid the overlapping of horizontal power or ground straps with the horizontal power or ground rails of standard cells. The default is `false`.

This option is mutually exclusive with the `-put_strap_in_std_cell_row` option.

Note:

Due to the alignment, the fixed strap pitch, fixed strap offset, and fixed power and ground spacing might not be honored, and some of the power and ground straps might be discarded due to the limited number of standard cell rows.

To prevent the core ring from blocking the standard cell rails, you can enable the "Keep rings outside of core area" option on the Global Power Network Constraints dialog box.

Placing Power and Ground Straps Inside Standard Cell Rows

You can place horizontal power and ground straps inside standard rows and between standard cell power or ground rails.

The syntax is

```
set_fp_rail_strategy -put_strap_in_std_cell_row true | false
```

When set to a value of `true`, power network synthesis places synthesized power or ground straps in the lowest horizontal layer inside standard cell rows and between standard cell power or ground rails. By using this strategy, you can avoid the overlapping of power or ground straps with the power or ground rails of standard cells even if they reside within the same logic net. The default is `false`.

This option is mutually exclusive with the `-align_strap_with_m1_rail` option.

Note:

To avoid rail blockages, the fixed strap pitch, fixed strap offset, and fixed power and ground spacing might not be honored, and some of the power and ground straps might be discarded due to the limited number of standard cell rows.

To prevent the core ring from blocking the standard cell rails, you can enable the "Keep rings outside of core area" option on the Global Power Network Constraints dialog box.

Performing Strap Alignment With Area I/O Cells

You can apply the following strategy if you want power network synthesis to synthesize the straps aligned with the pins of area I/O cells (bump cells) in a flip-chip design. You must specify the direction and metal layers of power and ground straps to be aligned with area I/O cells (bump cells) in a flip-chip design. Using this strategy can help save extra routing from the area I/O cell pins to the straps.

By default, power and ground straps are not aligned with area I/O cells.

The syntax is

```
set_fp_rail_strategy -align_strap_with_bump_cells  
align_constraint_file_name
```

Specifying Information in an Alignment Constraint File

Before power network synthesis can perform strap alignment with area I/O cells, you must also provide the following information in an alignment constraint file. You can specify multiple constraints.

The format of the alignment constraint file is:

```
net_name direction [width] [layer] [cell]
```

net_name is the name of the net to which the area I/O cell is aligned. (Required)

direction is the direction of the alignment straps, either horizontal or vertical. (Required)

width is the width of the alignment straps. By default, the width of other power network synthesis straps on the same layer is used. (Optional)

layer is the metal layer of the alignment straps. By default, the top metal layer in the same direction specified in the power network synthesis layer constraints is used. (Optional)

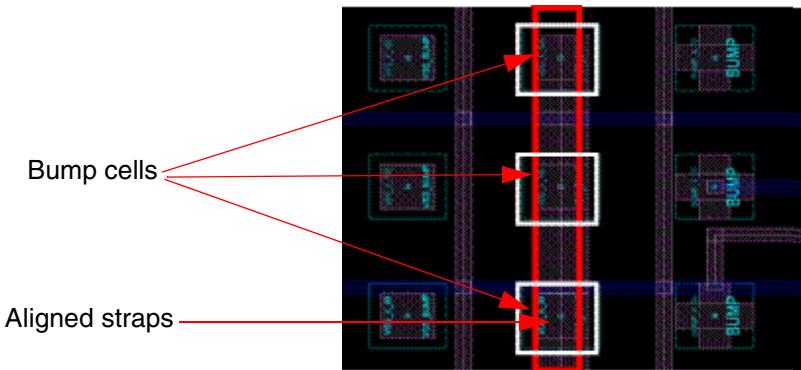
cell is the master name of the area I/O cell to be aligned. The default is all the area I/O cells in the design. (Optional)

Note:

When there is a DRC violation, the area I/O cell alignment straps have a higher priority than regular power network synthesis straps. However, in the case of multithreshold-CMOS alignment straps, the area I/O cell alignment straps have a lower priority.

[Figure 7-25](#) shows that the aligned strap width is the same as the bump cell pin width.

Figure 7-25 Strap Alignment With Bump Cells



Committing the Power Plan

Once the IR drop map meets the IR drop constraints, you can run the `commit_fp_rail` command to transform the IR drop map into a power plan. The `commit_fp_rail` command calls the `create_power_straps` command to generate the power net wires and vias.

Instead of running the `commit_fp_rail` command, you can click the Commit button in the Synthesize Power Network dialog box to actually generate a real power plan with net wires and vias.

Checking the Integrity of the Power Network

After you run power network synthesis you can check the integrity of your power and ground network early in the design planning stage by using the `check_fp_rail` command. Running an integrity check after you commit the power plan is important because it can reveal cases where the power network may not be exactly the same as the synthesized result. The integrity of your power network can be compromised by broken core rings or by floating segments that were left behind. If you have a multivoltage design with power-down voltage areas, the connection of the input pin of the power switch cell to the permanent power and ground straps may fail during the power network synthesis commit process.

The `check_fp_rail` command uses the following syntax:

```
check_fp_rail
-nets nets
-ring
-floating_segment
-power_switch_connection
```

Note:

The `check_fp_rail` command only checks the power and ground network at the design planning stage. It does not check the connection between the cell instances and the power straps, nor does it check standard cell rails.

- To specify the name of the power network on which to perform the integrity checking, use the `-net nets` option. You can specify a single net name or you can specify multiple net names separated by a comma or a space.
- To check for any broken core ring segments in the power network, use the `-ring` option. If any core ring segments are broken, a warning message is issued, identifying the location of the broken core ring segment in the power network.
- To check for floating wire segments in the power network, specify the `-floating_segment` option. The `check_fp_rail` command checks for both two-sided and one-sided floating wire segments. Two-sided floating wires do not connect to any power straps; one-sided floating wires connect to power straps on one side, but leave the other side unconnected.

If any floating segments are detected, a warning message is issued, identifying the location of the floating wire segments in the power network.

- To check the connection between the input pin on the multithreshold-CMOS power switch cells and the permanent power and ground straps, use the `-power_switch_connection` option. Before you use this option, specify the attributes of the multithreshold-CMOS switch cell on-resistance, the master cell name and layer of its primary and secondary power pins.

The command syntax is:

```
set_attribute [get_physical_lib_cells switching_cell_name]
"mtcmos_resistance" resistance_value

set_attribute [get_physical_lib_cells switching_cell_name]
"mtcmos_pin_layers" "pin1_name pin1_layer pin2_name pin2_layer"
```

If any unconnected power switch cell pins are detected, a warning message is issued, identifying the location of the unconnected switch cell pins in the power network.

Handling TLUPlus Models in Power Network Synthesis

Power network synthesis supports TLUPlus models for resistance effects on power and ground nets and CONN views.

By default, power network synthesis uses the sheet resistance defined in the technology file. However, TLUPlus models are used if one or both of the following conditions occurs.

- The USE_TLUPLUS environment variable is set. For example,

```
set_fp_rail_strategy -use_tluplus true
```

- The sheet resistance of any metal layer in the technology file is not defined.

If you use the TLUPlus model, power network synthesis outputs the following message to the log file:

```
Use TLU+ model for extraction
```

Specifying the Operating Temperature for IR Drop Analysis in Power Network Synthesis and Power Network Analysis

Because the metal resistance is dependent on operating temperature, you can specify a user-defined operating temperature to use for IR drop analysis in power network synthesis and power network analysis.

The syntax is

```
set_fp_rail_strategy -set_operating_temperature temperature
```

The *temperature* value is a floating number in the unit of Celsius degrees

Note:

The CRT coefficients must be defined in the TLUPlus model for the operating temperature to take effect. To do this, you must specify the `-use_tluplus true` option.

Supporting the Unified Power Format (UPF) Mode in Power Network Synthesis

Multivoltage power network synthesis supports the ability to automatically retrieve information on supply power nets and their voltages defined in the unified power format (UPF) for designs with multivoltage and multipower domains.

You must, however, specify the name of the voltage area based on the power domain that is associated with the voltage area. Power network synthesis can automatically set the supply net and voltage for the synthesis constraints. If nets are explicitly specified, then power network synthesis will also retrieve the voltage of that net from UPF.

Here is an example:

```
set_fp_rail_voltage_area_constraints
  -voltage_area VA1
  -power_budget 100.0

report_fp_rail_voltage_area_constraints
  -voltage_area VA1

---report voltage area VA1 synthesis constraints:
Voltage area name:          VA1
Net name:                   VDD1 VSS
Voltage supply [V]:          0.900
Power [mW]:                  100.000
Target Voltage Drop [mV]:    90.000
```

Note:

If you explicitly specify the supply net and voltage values for the voltage area(s) to be synthesized, and their values are different from the ones defined in the unified power format (UPF), power network synthesis will honor the input values you specified and issue a warning message.

Analyzing the Power Network

You can perform power network analysis to predict IR drop at different floorplan stages on both complete and incomplete power nets in your design.

Power network analysis consists of extraction and analysis of the power or ground rail specified by the given net names. You can analyze IR drop and electromigration effects during initial power grid planning while the floorplan with top-level blocks is still being implemented in the design and the power ports of the standard cells are not yet connected. Then later in the design cycle, when the details of top-level blocks have gone through detailed cell placement and power rail routing and connections have been established, you can verify whether power nets are of sufficient size and quantity.

For a selected hierarchy, power network analysis provides static rail analysis on power planned and on routed or unrouted designs as well as what-if analysis on virtual topological and voltage source changes.

This section includes the following topics:

- [Performing Power Network Analysis](#)
 - [Adding Virtual Pads](#)
 - [Ignoring Hard Macro Blockages When Connecting Virtual Pads to the Power Grid](#)
 - [Creating Connectivity \(CONN\) Views for Storing the Power and Ground Network of Hard Macros](#)
 - [Performing Power Analysis With Switching Activity Information](#)
 - [Handling a Long Via Array In Power Network Analysis](#)
-

Performing Power Network Analysis

To perform power network analysis,

1. Choose Preroute > Analyze Power Network.

The Analyze Power Network dialog box appears.

Alternatively, you can use the `analyze_fp_rail` command.

Set the options, depending on your requirements.

Power network analysis requires you to enter a power budget for the whole design.

- Specify the names of the power and ground nets on which to perform the power network analysis. All nets are processed simultaneously.
- Specify a total power budget for the analyzed design.

The power budget is divided among the instances, according to their size. The power unit is milliwatts. The default is 1,000.

For hard macros or standard cells, the power budget is computed based on the percentage of the total area.

For a hierarchical block, the power budget is computed based on the sum of all the cells and blocks inside it.

Note:

If there are no logical connections to the analyzed net, the allocation is zero.

Example:

If the power budget for the entire design is 500 milliwatts, the total area of the top-level blocks is 500,000 square microns, and one of the blocks has an area of 25,000 square microns, power network analysis allocates the power for that 25,000-square-micron block as follows:

$500\text{mW} * (25,000/500,000) = 25\text{mW}$

Note:

You can also obtain the power calculation source from a PWR view (for hard macros) or from Power Compiler and PrimePower report files.

- Specify a voltage supply by entering the voltage of the net you want analyzed. The unit is volts. The default is 1.5.
- Specify the power and ground pad information. The following power and ground pad definition types are supported:

Type 1: Specifying the input design pad file name – Enter the name of the pad file.

If the Instances option is active, you can enter a power and ground pad definition file in instance-based format.

The format is

instance_name net_name

(The *net_name* is optional.) For example,

VDD1 VDD

If the Masters option is active, you can enter a power and ground pad definition file in master-based format. This is the default.

The format is

master_name net_name

(The *net_name* is optional.) For example,

VDD.FRAM VDD

Type 2: Specifying pad masters – Enter the name of the pad master in the text box.

Type 3: Using top-level pins as pads – Select this option if you want pins to be regarded as pads during block-level power network synthesis.

- Specify a hierarchical option.

If your design contains macro cells that are placed and routed, you can sequentially search downward into these macro cells and bring all of the power and ground segments up to the top-level resistance network.

Flatten Hierarchical Cells (the default) – The power network analysis engine analyzes the design as if it were flat.

Top Level Cells Only – The power network analysis engine ignores cells inside soft macros but considers all the power and ground net wires (and vias) in the design.

- Specify an extraction option.

Create virtual rails – During the floorplanning stages, metal1 straps for the standard cell pin connections are usually not available. Power network analysis uses this option to create virtual pseudostraps that represent the metal1 straps for the standard cell pin connections. During power planning, it analyzes the IR drop and electromigration to predict what the effects might be when the metal1 straps are available.

Horizontal pseudostraps are created with specified layers for standard cell pin connections, based on the row information in the database.

Ignore CONN view layers – Reduces memory usage in power network analysis and power network synthesis and speeds up the IR simulation.

If you select the All option, all the metal layers in the CONN (connectivity) views are ignored.

If you select the Specified option, click the mask names of the layers to be ignored in the CONN views.

- Specify power information – There are four ways to obtain the power calculation source required for power network analysis:

Input power consumption file – To manually set power on a cell instance or cell master through a file, enter the name of the power consumption file.

Default – To manually enter a power budget, select the Default option and use the following power network analysis default format to obtain the power calculation:

<instance_or_master_name> power_value (mW)

Power Compiler – To obtain the power calculation source from a Power Compiler report file, select the Power Compiler option. After you complete a power calculation, use the following command syntax in Power Compiler to generate the report file:

`report_power -cell -flat -hier -nosplit report_file_name`

PrimePower/PrimeTime-PX – To obtain the power calculation source from a PrimePower or PrimeTime-PX report file, select this option. After you complete a power calculation in PrimePower, use the following command to generate the report file:

`calculate_power -waveform`

If `-waveform` is specified, PrimePower generates both instance peak current and average current information. Alternatively, use:

`report_power -file report_file_name -leaf`

The `-leaf` option is used for generating the power of all the leaf and block instances.

PrimeTime PX – After you complete a power calculation in PrimeTime PX, use the following command to generate the power file:

```
report_power -cell_power -leaf file_name
```

- Specify an output directory.

The power network analysis results are output to the default directory, pna_output. The specified directory stores the IR drop and electromigration results with the file name *design_name.net_name.pw_hl.pna* and the power allocation results with the file name *design_name.net_name.power*.

The IR drop and electromigration results are stored in the analysis file:

```
./pna_output/ designname.netname.pw_hl.pna
```

The power allocation results of the computed cells and blocks are stored in the analysis file:

```
./pna_output/ designname.netname.power
```

Note:

Because these files can be quite large (more than 250 MB for 500K instances), make sure you have enough disk space in your ./pna_output directory.

2. Click OK or Apply.

Adding Virtual Pads

The power network analysis engine assumes that the voltage source comes from power pads. Power network analysis enables you to insert virtual voltage sources by adding virtual pads at different x- and y-locations without actually adding or relocating power pads. If the location is outside of the power net wires and vias, the power network analysis engine finds the closest neighboring wires from which to attach the virtual pads.

Virtual pad parameters are stored in a temporary memory buffer. Power network analysis does not add these pads to the design database. Virtual pad parameters are valid only for the current analysis session.

You can perform what-if analysis to see whether improvements can be made to the power grid design. In addition, you can significantly reduce voltage drop effects, by changing the number of pads used, pad placement, or both the number of pads and the location.

To add virtual pads,

1. Choose Preroute > Create Virtual Power Pads.

The Virtual Power Pads dialog box appears.

Alternatively, you can use the `create_fp_virtual_pad` command.

2. Set the options, depending on your requirements.

- To create the virtual voltage pad, enter the name of the power or ground net, such as VDD or VSS. Only a single net is allowed at a time.
- Specify the layer name of the virtual pad you want to create.

If you select Auto (the default), the topmost routing layer that has power and ground wires intersecting the x- and y-locations automatically determines the layer of the virtual voltage pad.

If you select Specified, select the name of the metal layer on which to create the virtual voltage pad.

- Specify a virtual pad point (x, y) that you want to place.
- Add to the virtual power pads list as needed.

This list displays the names of all the selected virtual pads that are to be incorporated into the power network analysis. To add a virtual pad to the list, click where you want the pad to be in the layout window, select a net and layer, and then click Add to add it to the list.

- Specify a virtual power pad name.

Enter the name of the virtual pad file that you want to add to the virtual power pads list. The format for the virtual pad files is

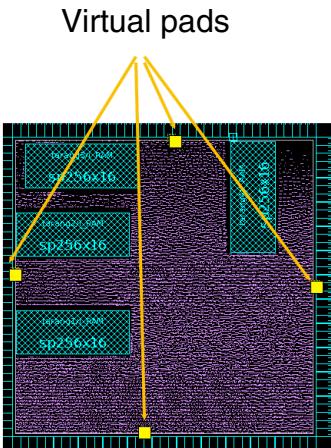
```
<net_name>
<x_coord> <y_coord> <layer_name>
```

Load – The virtual pad parameters are loaded into a temporary memory buffer. Power network analysis does not add these pads to the design database. Virtual pad parameters are stored in a temporary memory buffer that is valid only for the current analysis session.

Save – Enter the name of the virtual pad file you want to save.

3. Click Close.

[Figure 7-26](#) shows the virtual pads.

Figure 7-26 Virtual Pads Example

Ignoring Hard Macro Blockages When Connecting Virtual Pads to the Power Grid

By default, the virtual pad connection to the power grid considers the presence of hard macro blockages. If the connecting path is blocked by hard macro instances, the virtual pad is ignored.

You can ignore the hard macro blockages. To do this, select Preroute > Power Network Constraints > Global Constraints to display the Global Power Network Constraints dialog box. Select the “Ignore all the blockages” option.

Alternatively, you can use the `set_fp_power_plan_constraints` command.

Removing Virtual Pads

You can remove (delete) virtual power or ground pads by using the `remove_fp_virtual_pad` command. You can remove the pads by specifying the names of the power or ground nets, the layer name of the pad, or the virtual pad point. You can also remove all the virtual pads in the design.

The `remove_fp_virtual` command uses the following syntax:

```
remove_fp_virtual_pad [-nets string] [-layer string] [-point {x y}] [-all]
```

Creating Virtual Connections From Power and Ground Pads to Power and Ground Wires In Power Network Analysis

You can enable power network analysis to make virtual connections from power and ground pads to power and ground wires based on predefined connections specified in a configuration file by using the `-define_pad_connection connection_config_file_name` option with the `set_fp_rail_strategy` command. The configuration file includes the master or instance name of the pads, connected pin layers, and routing layers.

The format of the connection configuration file is:

```
Instance | Master name
```

```
Route Pins layer layer_number | all
```

```
Primary Routing Layer Preferred low | high
```

or

```
Primary Routing Layer Specified Horizontal layer_number
```

```
Primary Routing Layer Specified Vertical layer_number
```

- The pad instance or master name specifies the instance or master name of the pads.
- The connected pin layers connect pins on all layers or on a specified layer.
- The primary routing layer specifies the layer on which to perform the routing: preferred, pin, or specified. You can route on preferred metal layers or on the next available higher or lower preferred layer. You can route on the same layer on which the pins are located, and you can define the horizontal or vertical routing layers.

Creating Connectivity (CONN) Views for Storing the Power and Ground Network of Hard Macros

Nonstandard cells, such as hard macros, can have significant effects on voltage drop and electromigration effects at the full-chip level. The power consumption of these hard macro cells or the power and ground routing that exists inside the cells might allow current to flow through the hard macro cells and cause voltage to be applied to other regions of the chip. It is necessary, therefore, to be able to consider the internal power and ground routing and power consumption of these hard macro cells during top-level rail analysis.

The hard macro flow in power network analysis allows you to create a combination of the routing and power usage values into a single construct that power network analysis (`analyze_fp_rail` command) can access at the full-chip level. The construct consists of two parts:

- A connectivity (CONN) view that represents, in a viewable manner, the power and ground networks inside the hard macro cell(s) in question.
- A current source file (CSF) in the ASCII format that records where current is drawn inside the hard macro cell.

Use the `create_connview` command to invoke the Hercules connectivity engine automatically to create CONN views and current source files for the hard macro cells. Power network analysis uses the `create_connview` command options and values you specify, along with those specified in the Milkyway library's technology file, to create a Hercules runset file, called `runset.ev`. The Hercules connectivity engine then uses the runset file and creates a SMASH view in the database. After the SMASH view is created, the command uses this data to create a CONN view for the hard macro cell(s) in question.

Power network analysis updates the runset file every time you run the `create_connview` command.

Note:

Before you start power network analysis, make sure Hercules is installed in the path where Power Network Analysis is installed.

The `create_connview` command uses the following syntax:

```
create_connview
[-power_nets net_names]
[-ground_nets net_names]
[-generate_csf]
[-skip_via mask_names]
[-connview_skip_cell cell_names]
[-csf_skip_cell cell_names]
[-layer_text mask_names_text]
-design design_name
-library library_name
```

Before you run the `create_connview` command, close the library and cell of interest. The number of cells printed in each Hercules command is limited to 300 to prevent the Hercules parser from breaking.

Specifying the Design and Library Names

You can specify the name of the cell for which you want to generate the CONN view by using the `create_connview -design design_name` option.

Note:

Because the `create_connview` command runs on the CEL view of the specified cell rather than on GDSII data, the cell in question must exist within the Milkyway database.

You can specify the name of the library that contains the cells for your design by using the `create_connview -library library_name` option.

Specifying the Hierarchical Power and Ground Net Information for Extraction

You can specify the hierarchical power and ground net information by using the `create_connview -power_nets` and `-ground_nets` options and entering the names of the top-level power and ground nets to be extracted.

Generating Current Source Files During CONN View Generation

If you want to generate current source files (CSFs) in an ASCII format and attach them to the CONN view during CONN view generation, use the `create_connview -generate_csf` option. The two major parts in the current source file are the current source locations and the current values. The current source locations can be determined by the locations of diffusion contact cuts to the approximate MOS transistor terminal locations.

You must specify the name of the cell for which you want to generate the current source files, and the metal-to-metal intersections to use for the current source locations. These are the intersections in the power and ground network for the individual nets. The generated current source file is net-based, meaning that each power and ground net has a corresponding current source file.

When the `create_connview -generate_csf` option is executed, power network analysis reports that it is processing geometry for each of the power and ground nets in the CONN view of the hard macro cell. For example, if the hard macro cell name is ram, the created source files are as follows:

```
ram:1  
ram:1_1  
ram:1_2  
...  
ram:1_#
```

`ram:1` is the view of the power and ground geometries used by the tool when you want to open and view the CONN view of the cell (`ram.CONN`)

`ram:1_1`, `ram:1_2`, ..., and `ram:1_#` are the text files that store the locations of the metal-to-metal intersections in the power and ground network.

Excluding Vias During CONN View Generation

You can specify the mask names of the specified vias you want the connectivity engine to exclude when it traces the connectivity of a multi-layer pin and extracts the entire pin by using the `create_connview -skip_via mask_names` option. The default is to extract the connectivity through via1 through via11 (metal1 to metal12).

Excluding Cells During CONN View and Current Source File Generation

You can specify the child cells to be “skipped” (excluded) from the current CONN view generation by using the `create_connview -connview_skip_cell cell_names` option. You must specify the names of the child cells. By default, CONN views are generated for all the child cells in the design.

You can specify the child cells to be “skipped” (excluded) from the current source file generation by using the `create_connview -csf_skip_cell cell_names` option. By default, current source files are generated for all the child cells in the design.

The “skip cell” feature reduces the data size and improves the performance of the CONN view generation process without losing the accuracy of the IR drop analysis.

Because the connectivity engine cannot distinguish the cell types from GDSII data, the “skip cell” feature is available only to standard cells. For other cell types, current sources are snapped to the cell boundaries only when the center points of the power and ground pin rectangles are inside the cell boundaries.

Note:

If you are creating a full-chip CONN view or an IP block CONN view for rail analysis (`analyze_fp_rail` command), it is sometimes better not to generate CONN views for standard cells to remove the power and ground fingers. After the fingers are removed, the current source files located at the fingers are snapped to the center of standard cell power and ground pins. By “skipping” bit cells, memory usage is saved in the downstream processing steps.

- Skipping CONN view generation but not CSF generation – If you select the `-connview_skip_cell` option and specify a cell, but not the `-csf_skip_cell cell_names` option, the tool does not generate the CONN views for the specified cell but the current sources at the diffusion contacts in the cell are snapped to the center of the cell instance boundaries that interact with the top-level power and ground pins.

Note that this setting applies to standard cells only.

- Skipping CSF generation but not CONN view generation – If you select the `-csf_skip_cell` option and specify a cell, but not the `-connview_skip_cell` option, the tool generates only the cell content from the CONN views for the specified cell; no current source file is generated.

Note that this setting usually applies to I/O pad cells.

Identifying the Geometry Annotated By the Text

If you want power network analysis to view the layers beneath the origin of each text label to identify the corresponding metal layer geometries annotated by the text, use the `create_connview -layer_text mask_names_text` option. Specify the metal mask names with the name or number of the layers used for each text label to identify the geometries.

The format of the layer text is:

`mask_name:layer_name_or_number`

For example,

`-layer_text {metal2:32 metal3:31}`

In this example, layer 32 is used to identify metal2 geometry, and layer 31 is used to identify metal3 geometry.

Note:

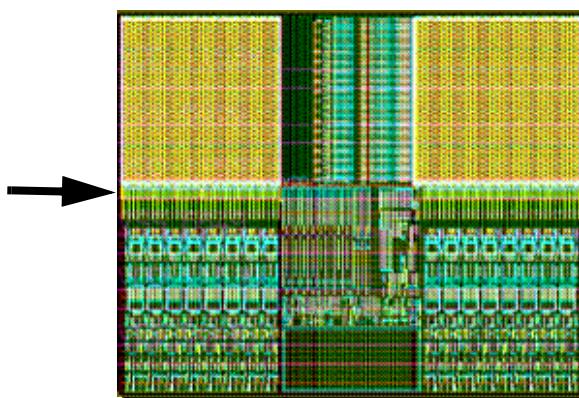
When text layers are not specified, power network analysis assumes they match the geometry layers.

[Figure 7-27](#) shows an example of CONN view generation.

Figure 7-27 Example of CONN View Generation

```
create_connview -library ts1ge256x16m4_lib  
-design ts1ge256x16m4 -power_nets VDD  
-ground_nets VSS
```

CONN View



Performing Power Analysis With Switching Activity Information

The power of a design depends on the switching activity of the design nets and cell pins. The more accurate the switching activity, the more accurate the power analysis. To calculate the power, you must first annotate the design with switching activity information. Switching activity can be annotated on design nets, ports, and cell pins.

After capturing the switching activity and annotating your design, you can invoke power network analysis by using the `analyze_fp_rail -analyze_power` or `synthesize_fp_rail -analyze_power` command to calculate the dynamic power values of each cell instance from IC Compiler by considering specified switching probabilities and using the obtained power values for IR drop analysis. (See “[Performing Power Network Analysis for Each Cell Instance](#)” on page 7-110.)

Annotating Switching Activity

You can annotate the switching activity with one of the following methods:

- You can use the `set_switching_activity` command to annotate particular design objects, such as nets, pins, ports, and cell instances of the current design.

Switching activity information consists of the static probability and the toggle rate. The static probability is the probability that the value of the design object has a logic value 1. It defines the percentage of time the signal is at a high state. By default, the value is 0.5, which means the signal is high for half the time and low for half the time. The toggle rate is the rate at which the design object switches between logic values 0 and 1 within a time period.

The `set_switching_activity` command uses the following syntax:

```
set_switching_activity
  [-static_probability sp_value]
  [-toggle_rate tr_value]
  [-state_dep state_condition]
  [-path_dep path_sources]
  [-rise_ratio ratio_values]
  [-period period_value | -clock clock_name]
  [-select select_types]
  [-hier]
  [-instances instances]
  [-object_list]
  [-verbose]
```

For example,

```
set_switching_activity -toggle_rate 0.25 -period 10 -clock CLK
  -static_probability 0.015 [all_inputs]
```

In this example, the `set_switching_activity` command annotates a toggle rate value of 0.25, and a static probability value of 0.015 to all design input ports. The clock The clock, CLK, is specified as the related clock for all inputs.

When power is calculated, the annotated toggle rate of 0.25 is divided by the clock period (10) and the resulting toggle rate (0.25) is used.

For more information about the `set_switching_activity` options, see the man page.

- You can generate one or more switching activity interchange format (SAIF) files using RTL or gate-level simulation and use the `read_saif` command to annotate the switching activity information onto nets, pins, ports, and cells in the current gate-level design.

A SAIF file is usually generated using a simulation flow, where a simulation testbench instantiates the design being simulated and provides simulation vectors. The generated SAIF file contains the switching activity information organized in a hierarchical fashion, where the hierarchy of the SAIF file reflects the hierarchy of the simulation testbench.

The `read_saif` command uses the following syntax:

```
read_saif
  -input file_name
  -instance_name name
  [-target_instance instance]
  [-ignore ignore_name]
  [-ignore_absolute ig_absolute_name]
  [-exclude exclude_file_name]
  [-exclude_absolute ex_absolute_file_name]
  [-names_file_name_changes_log_file]
  [-scale scale_value]
  [-unit_base unit_value]
  [-khrate khrate_value]
  [-map_names]
  [auto_map_names]
  [-rtl_direct]
  [-verbose]
```

For example,

```
read_saif -input file1 -instance_name Test/U1
```

In this example, the `read_saif` command annotates the switching activity on the current design, which was instantiated as *Test/U1* in the simulation testbench.

For more information about the `read_saif` options, see the man page.

- You can use the default switching activity.

If no simulation data is available, IC Compiler applies the built-in default toggle rate to analyze the power consumption.

The switching activity of primary inputs and black box outputs cannot be propagated. You can set the default toggle rate (0.1) and default static probability (0.5) for primary inputs and black box outputs by using the "set_power_default_toggle_rate" and "set_power_default_static_probability" variables. The default toggle rate value is the value of the power_default_toggle_rate multiplied by the related clock frequency. The related clock can be specified by using the -clock option of the set_switching_activity command. If no related clock is specified on the net, the clock with the highest frequency is used.

The default switching activity applied to these objects is then propagated throughout the design for power network analysis.

Performing Power Network Analysis for Each Cell Instance

After capturing the switching activity and annotating your design, you can invoke power network analysis by using the `analyze_fp_rail -analyze_power` or `synthesize_fp_rail -analyze_power` command to analyze the power information from each cell instance in your design based on static probability and toggle rates. It calculates the dynamic power values of each cell instance from IC Compiler by considering specified switching probabilities and uses the obtained power values for IR drop analysis. By default, power is not calculated from IC Compiler unless the `-analyze_power` option is explicitly used.

For example,

```
analyze_fp_rail -analyze_power -nets {VDD} -voltage_supply 1.2
```

In this example, the voltage drop on net VDD is analyzed by IC Compiler and the voltage supply is 1.2 volts.

You can use the `-analyze_power` option concurrently with the `-power_budget` and the `-read_default_power_file` options.

If you specify a power budget with the `-analyze_power` option and the power budget is larger than the sum of the cell instances calculated by `-analyze_power`, the difference between the power budget and the total power calculated by `-analyze_power` will be assigned to those instances, if there are such instances, that are not assigned in `-analyze_power`. Otherwise, the power budget is ignored. By default, the power budget is set to 0 with the `-analyze_power` option.

If you use the `-analyze_power` option concurrently with the `-read_default_power_file` option, the power specified in the default power file has a higher priority than that calculated by the `-analyze_power` option. This means that if the power value is calculated and read from the default power file for one instance, the instance will be assigned with the power value from the default power file.

Handling a Long Via Array In Power Network Analysis

Power network analysis can break a large square or a rectangular via into smaller square via arrays when performing power and ground network extraction. Each via will be represented by several resistors, depending on the cutting rule you specify.

The syntax is

```
set_fp_rail_strategy -pna_via_cut_row_column number
```

where *number* can be any integer.

For example, if you define the size as 5 and the design has one 1 x 10 via array, one 10 x 1 via array, and one 9 x 9 via array, the cutting results are as follows:

Original Via Layer	After Cutting
1 x 10 via array	Two 1 x 5 via arrays
10 x 1 via array	Two 5 x 1 via arrays
9 x 9 via array	One 5 x 5 via array, one 4 x 5 array, one 5 x 4 via array, and one 4 x 4 array

Viewing the Analysis Results

When power and rail analysis are complete, you can check for the voltage drop and electromigration violations in your design by using the voltage drop map and the electromigration map. You can also detect the power and ground weaknesses that a voltage drop map cannot by using a resistance map.

You can save the results of voltage drop and electromigration current density values to the database by saving the CEL view that has just been analyzed. Doing so allows you to close the database and retrieve the results later.

Displaying the Voltage Drop Map

After rail analysis is complete, you can check for voltage drop distribution in your design. A voltage drop map produced by power network analysis can also be used as a reference IR drop map to detect subsequent power routing problems.

The tool performs voltage drop analysis by retrieving the database objects that belong to the power and ground net to be analyzed and then builds a resistive network through parasitic extraction. All cell instances are treated as equivalent current sources, and their value is derived from the power consumption values you specify.

Voltage drop violations are displayed in a highlight map, which provides a graphical view of the results.

To display the voltage drop violations in a highlight map,

1. Choose Preroute > Power Network Voltage Drop Map.

The PNA Voltage Drop Map Mode panel appears.

Alternatively, you can use the `load_fp_rail_map` command.

2. Select metal layers for analysis, and enter the maximum and minimum threshold values.

By default, power network analysis automatically sets the upper and lower bounds of the net you specify to the calculated minimum and maximum threshold values.

3. Click Reload.

The Load Voltage Drop Map Data dialog box appears.

4. Select the net for which you want to load the rail analysis results for displaying the voltage drop map.

5. Click OK to load and display the IR drop map in the current layout view.

A voltage drop map divides the core area into a grid of colored boxes. Each box represents the voltage drop levels in that area of the design. The box edges are colored and labeled to show the voltage drop values. Each map color represents a range of voltage drop values called a bin.

The voltage drop map legend also displays the color and the data count. The colored histogram bars on the right side of the legend represent the relative distribution of voltage drop values in the bins.

To view voltage drop data in the map by using the Query tool,

1. Click the Query button on the Map Mode panel to enable the Query tool.

The Query Object panel appears.

2. Click a location in the map for which you want to view the voltage drop data.

The voltage drop data is displayed on the Query Object panel. The object is highlighted in the query color (white) and the information for the object appears on the Query Object panel.

Displaying the Electromigration Map

After rail analysis is complete, you can check for electromigration violations in your design. Because different metal and via layers usually have different threshold current densities, you can set independent maximum and minimum display values for clearly conveying the difference in threshold current densities.

To display the electromigration violations in a highlight map,

1. Choose Preroute > Power Network Electromigration Map.

The PNA Electromigration Map Mode panel appears.

Alternatively, you can use the `load_fp_rail_map` command.

2. Select metal layers for analysis, and enter the maximum and minimum threshold values.

By default, power network analysis automatically sets the upper and lower bounds of the net you specify to the calculated minimum and maximum threshold values.

3. Click Reload.

The Load Electromigration Map Data dialog box appears.

4. Select the net for which you want to load the rail analysis results for displaying the electromigration map.

5. Click OK to load and display the electromigration map in the current layout view.

An electromigration map divides the core area into a grid of colored boxes. Each box represents the electromigration levels in that area of the design. The box edges are colored and labeled to show the electromigration values. Each map color represents a range of electromigration values called a bin.

The electromigration legend also displays the color and the data count. The colored histogram bars on the right side of the legend represent the relative distribution of electromigration values in the bins.

To view electromigration data in the map by using the Query tool,

1. Click the Query button on the Map Mode panel to enable the Query tool.

The Query Object panel appears.

2. Click a location in the map for which you want to view the electromigration data.

The electromigration data is displayed on the Query Object panel. The object is highlighted in the query color (white) and the information for the object appears on the Query Object panel.

Displaying the Resistance Power Map

After performing rail analysis, you can display a new Power Network Analysis Resistance map where each node and edge on the power and ground network has a color corresponding to its resistivity value. The resistivity value of a node is the resistance value on a signal path that is the smallest resistive path from the node to one of the design's boundary nodes. You can use this map to check for resistance distribution in your design. You can display the resistance map when rail analysis is complete.

The resistance power map can detect the power and ground weaknesses that a voltage drop map cannot. When the resistance map is displayed, you can check for the weaknesses of the power and ground network. The weaknesses to be checked include missing vias, discontinuous connections, and insufficient vias.

The resistance map shows the minimum resistance from ideal voltage sources to any given point in the network. Thus, when you query regarding a net segment on the resistivity map, the value returned is the minimum resistance from an ideal voltage source to this segment.

To display the resistance power map,

- Choose Preroute > Power Network Resistance Map.
The PNA Resistance dialog box appears.
Alternatively, you can use the `load_fp_rail_map` command.

The first time you display a resistance map during a session, you must load the resistance map data. You can then reload the data if it changes during a session.

To load or reload the resistance map data,

- Enter the name of the power or ground net for which you want to load the resistance values in the design.
- Set the maximum and minimum thresholds by selecting the "Max threshold" and "Min threshold" options.

By default, power network analysis automatically sets the upper and lower bounds of the net you specify to the calculated minimum and maximum threshold values.

- Click the Reload button.

The Resistance Map Data dialog box appears.

- Select the name of the power or ground net that you want to analyze.
- Type (or browse for) the name of the directory where the rail analysis data is located in the "Input directory" box.

6. Click OK.

A resistance map divides the core area into a grid of colored boxes. Each box represents the resistance values in that area of the design. The box edges are colored and labeled to show the resistance values. Each map color represents a range of resistance values called a bin. The default is 10.

To query resistance map data,

1. Click the Query Tool button.
2. Click on a power or ground net node location. The tool will list the shortest resistance path from the selected instance to the closest pad in the command window, as well as the information about all the resistors on this path.

Displaying Instance Voltage Drop, Power, and Power Density Maps

After you run power network analysis or power network synthesis, you can load instance voltage drop, power, and power density maps for display in IC Compiler.

Displaying the Instance Voltage Drop Map

After performing preroute rail analysis, you can use the instance voltage drop map to check for voltage drop distribution. Since the voltage drop map is produced from power network analysis, you can use it as a reference IR drop map to detect subsequent power routing problems.

To display the instance voltage drop map, the GUI retrieves the data objects and builds a resistive network through parasitic extraction. All cell instances are treated as equivalent current sources.

To display an instance voltage drop map for the current design,

1. Choose Preroute > Power Network Instance Voltage Drop Map.

The PNA Instance Voltage Drop dialog box appears.

Alternatively, you can use the `load_fp_rail_map -instance -type InstIR` command.

The GUI dims the visible objects in the layout view and displays the Map Mode panel. The first time you display an instance voltage drop map during a session, you must load the voltage drop map data. You can reload the data if it changes during the session.

2. Set the maximum and minimum threshold values for the instance-based voltage drop limits. Enter the values in the “Max threshold” and “Min threshold” boxes.

By default, the values are automatically set, based on the power rail synthesis or power rail analysis results from a specified directory.

3. Click the Reload button on the map mode panel.

The Load Instance Voltage Drop Map appears.

4. Type or browse for the name of the directory where the instance IR voltage drop data is located in the Input directory box. The default output directory is pna_output. For the instance based voltage drop map, the file name is *design_name.inst_hl.pna*.

5. Click the Query button to query the instance voltage drop data.

6. Click OK.

The instance voltage drop map divides the core area into a grid of colored boxes. Each box represents the voltage drop levels in that area of the design. The box edges are colored and labeled to show the voltage drop values. Each map color represents a range of values called a bin.

Displaying the Instance Power Density Map

You can use the instance power density map to examine power density levels.

To display an instance power density map,

1. Choose Preroute > Power Network Instance Power Density Map.

The PNA Power Density dialog box appears.

Alternatively, you can use the `load_fp_rail_map -instance -type PD` option.

The GUI dims the visible objects in the layout view and displays the Map Mode panel. If you have already generated instance power data during the session, the instance power map grid appears on top of the design in the layout view. If the map does not appear, you must load the power data. You can reload the data if it changes during the session.

2. Set the maximum and minimum threshold values for the instance-based voltage drop limits. Enter the values in the “Max threshold” and “Min threshold” boxes.

By default, the values are automatically set, based on the power rail synthesis or power rail analysis results from a specified directory.

3. Click the Reload button on the map mode panel.

The Load Instance Power Density Map Data dialog box appears.

4. Type or browse for the name of the directory where the instance power density data is located in the Input directory box. The default output directory is pna_output. For the instance power density map, the file name is *design_name.inst_hl.pna*.

5. Click the Query button to query the instance power density data.

6. Click OK.

The instance power density map divides the core area into a grid of colored boxes. Each box represents the power density levels on a vertical plane and a horizontal plane. The left and bottom box edges are colored and labeled to show the power density values for the planes. Each map color represents a range of values called a bin.

Displaying the Instance Power Map

You can use the instance power map to examine power levels.

To display an instance power map,

1. Choose Preroute > Power Network Instance Power Map

The PNA Instance Power dialog box appears.

Alternatively, you can use the `load_fp_rail_map -instance -type P` option.

The GUI dims the visible objects in the layout view and displays the Map Mode panel. If you have already generated instance power data during the session, the instance power map grid appears on top of the design in the layout view. If the map does not appear, you must load the power data. You can reload the data if it changes during the session.

2. Set the maximum and minimum threshold values for the instance-based voltage drop limits. Enter the values in the “Max threshold” and “Min threshold” boxes.

By default, the values are automatically set, based on the power rail synthesis or power rail analysis results from a specified directory.

3. Click the Reload button on the map mode panel.

The Load Instance Power Map Data dialog box appears.

4. Type or browse for the name of the directory where the instance power data is located in the Input directory box. The default output directory is `pna_output`. For the instance power map, the file name is `design_name.inst_h1.pna`.

5. Click the Query button to query the instance power map data.

6. Click OK.

The instance power map divides the core area into a grid of colored boxes. Each box represents the power levels on a vertical plane and a horizontal plane. The left and bottom box edges are colored and labeled to show the instance power values for the planes. Each map color represents a range of values called a bin.

Summary of Power Network Analysis and Power Network Synthesis Options

Table summarizes the power network analysis and power network synthesis options. Use the `set_fp_rail_strategy` command to set the following options. The option settings are not persistent in the Milkyway database.

Variable	Usage
<code>-reset</code>	Resets the specified strategy value. If no strategy name is specified, all the strategy values will be reset.
<code>-use_lm_view true false</code>	Uses the Logic Modeling (LM) view for power calculations. The default is false.
<code>-pna_ultra_solver true false</code>	Enables power network analysis to use the Ultra Solver which uses less memory but more CPU time. Ultra Solver is a high-capacity and memory-efficient solver ideal for extremely large and dense regular power meshes. Unlike the regular solver, it approximates IR/EM solution within a certain error tolerance. For power meshes with an irregular topology and density, this might take longer than the regular solver, and therefore, it is recommended that you turn this variable on (true) when the regular power network analysis and power network synthesis sessions run out of memory. The default is false.
<code>-virtual_pad_wire_width distance</code>	Defines the width of the virtual wires used to connect the virtual pads and their neighboring wires. By default, the power network synthesis and power network analysis virtual pad uses the closest wire width for a virtual connection.

Variable	Usage
<code>-pad_resistance_file <i>file_name</i></code>	Specifies the file from which to read in the pad resistance. By default, the tool reads the pad resistance from the technology file.
<code>-pns_skip_ir true false</code>	By default, power network synthesis always performs IR drop simulation before committing the power mesh, even when the number or width of the power and ground straps is fixed. When this option is set to true, power network synthesis skips IR drop simulation.
	The default is false.
<code>-pns_clip_top_boundaries true false</code>	When this option is set to true, power network synthesis turns on the “clip at top-cell boundaries” option in the preroute engine, when committing the power networks.
	The default is false.
<code>-pns_ignore_via_cut_to_edge true false</code>	When option is set to true, the synthesized power and ground wires will ignore the via’s cut to edge spacing. This strategy might result in a DRC violation.
	The default is false.

Variable	Usage
-pns_hor_relative_offset <i>layer distance</i>	By default, power network synthesis overlaps straps with the same directional layers. However, this may result in problems such as unexpected stacked vias for certain designs.
-pns_ver_relative_offset <i>layer distance</i>	Use these two parameters to set the relative offset from the topmost layer (horizontal or vertical) to the specified layer. This allows power network synthesis to synthesize non-overlapped straps with the same directional layers.
	There are two arguments for this option. The first is the layer name, and the second is the offset (in microns) from the topmost layer to the specified layer.
	For example, suppose the topmost horizontal layer is metal5. In this case, you can specify the following:
	PNS_HOR_RELATIVE_OFFSET "metal3 15"
	It specifies the offset from the top edge of metal3 for power nets to the top edge of metal5 for power nets to 15 microns. If the offset is a positive value, the metal3 straps will be at the top of the corresponding metal5 straps, and if the offset is a negative value, the metal3 straps will be at the bottom of the corresponding metal5 straps.
-pns_ignore_soft_macro_blockage true false	When this option is set to true, soft macro blockages are ignored.
	The default is false.

Variable	Usage
<code>-pns_commit_block_ring_first true false</code>	When this option is set to true, power network synthesis commits the block ring before it commits the power straps.
	The default is true.
<code>-set_operating_temperature temperature</code>	Use this option to specify a user-defined operating temperature to use for IR drop analysis in power network synthesis and power network analysis.
	Note that for the temperature setting to take effect, you must also specify the <code>-use_tluplus true</code> option.
<code>-use_tluplus true false</code>	When this option is set to true, power network synthesis uses the TLUPlus model.
	The default is false.
<code>-pna_via_cut_row_column number</code>	Use this option to break a large square or a rectangular via array into smaller square via arrays, based on the specified value.
<code>-honor_macro_route_constraints file_name</code>	Use this option to specify user-defined power and ground routing constraints for hard or soft macros or both in an input file. The file defines the macro type, the name of the macro, the name of the net that will be blocked, and the name of the layer that will block the power and ground routing.
<code>-honor_macro_strap_config configuration_file</code>	Enables power network synthesis to accept user-defined power and ground configurations containing layer, direction, width, pitch, and offset on specified hard macros or particular rectangular regions. The remaining area is synthesized based on the user-defined constraints.

Variable	Usage
<code>-align_strap_with_top_pin true false</code>	Use this option to specify whether or not to align power and ground straps with all the top-level power and ground pins. If this option is set to a value of <code>true</code> , power network synthesis aligns the power and ground straps to all the power and ground pins on the block boundary that are connected to the net of the same type as the straps. This strategy is useful when performing block-level power network synthesis.
	The default is <code>false</code> .
<code>-align_strap_with_mtcmos_cells file_name</code>	Use this option to automatically enable power network synthesis to align power and ground straps with specified multithreshold-CMOS power switch cells in the design and to make connections to multithreshold-CMOS pins. This is useful in a multithreshold-CMOS design flow where the power switching cells have been placed in the design before the power network is created.
<code>-align_strap_with_bump_cells file_name</code>	Use this option to specify the direction and metal layers of power and ground straps to be aligned with area I/O cells in a flip-chip design.
<code>-align_strap_with_m1_rail true false</code>	When set to a value of <code>true</code> , power network synthesis aligns synthesized power and ground straps in the lowest horizontal layer with the horizontal power or ground rails of standard cells. By using this strategy, you can avoid the overlapping of horizontal power or ground straps with the horizontal power or ground rails of standard cells.
	The default is <code>false</code> .
	This option is mutually exclusive with the <code>-put_strap_in_std_cell_row</code> option.

Variable	Usage
<code>-put_strap_in_std_cell_row true false</code>	<p>When set to a value of <code>true</code>, power network synthesis places synthesized power or ground straps in the lowest horizontal layer inside standard cell rows and between standard cell power or ground rails. By using this strategy, you can avoid the overlapping of power or ground straps with the power or ground rails of standard cells even if they reside within the same logic net.</p>
	<p>The default is <code>false</code>.</p> <p>This option is mutually exclusive with the <code>-align_strap_with_m1_rail</code> option.</p>
<code>-define_pad_connection file_name</code>	<p>Use this option to enable power network analysis to make virtual connections from power and ground pads to power and ground wires based on predefined connections specified in a configuration file.</p>
<code>-commit_fast true false</code>	<p>When set to a value of <code>true</code>, power network synthesis calls the <code>create_power_straps</code> command once for each layer when committing the synthesized power plan (<code>commit_fp_rail</code> command).</p>
	<p>The default is <code>false</code>, meaning that power network synthesis calls the <code>create_power_straps</code> command once for each segment when committing the synthesized power plan.</p>

Variable	Usage
<code>-cut_plangroup_edge_layers true false</code>	<p>When a plan group is committed into a soft macro and the top-level power and ground wires are pushed down into the block, any power and ground wires that overlap with the edge of the plan group, particularly at the lower metal layers, may block the assignment of the signal pins inside the soft macro. To solve this issue, set this parameter to a value of <code>true</code> to cut the synthesized straps that overlap with the plan group edges.</p> <p>The default is <code>false</code>.</p>
<code>-create_hierarchical_pns_script true false</code>	<p>When set to a value of <code>true</code>, a subdirectory called <code>plangroup_name_PNS_script</code> is created under the <code>pna_output</code> directory with a name that is associated with each plan group in the directory. The following two files are generated from this subdirectory:</p> <p><code>plangroup_name.vin</code> which contains the voltage information for each soft macro pin on the plan group boundary.</p> <p><code>plangroup_name.replay.tcl</code> which resynthesizes the PG mesh in the soft macro.</p>

Reporting Settings for Power Network Synthesis and Power Network Analysis Strategies

You can get a report of the current values of the strategies used by power network synthesis and power network analysis by using the `report_fp_rail_strategy` command.

Running PrimeRail Within IC Compiler

Aside from running PrimeRail as a stand-alone tool, you can execute PrimeRail commands within IC Compiler for power network resistivity checking, voltage drop analysis, and electromigration analysis.

This section includes the following topics:

- [IC Compiler—PrimeRail Analysis Flow](#)
 - [Debugging IC Compiler—PrimeRail Flow](#)
 - [Calculating Net Resistivity](#)
 - [Exporting PrimeRail Analysis Data for IC Compiler](#)
 - [Using the create_rail_setup Command to Perform Reliability Analysis](#)
-

IC Compiler—PrimeRail Analysis Flow

Reliability analysis can be done after various stages of IC Compiler are completed, such as placement, clock tree synthesis, global routing, detailed routing, or chip finishing. It is important to catch and resolve most of the power network issues as early as possible in the design cycle. To extend the usability and effectiveness of sign-off rail analysis, you can invoke the PrimeRail tool in batch mode within IC Compiler if you want to complete power network checking and analysis at any stage, starting with the end of placement and clock tree synthesis.

When the reliability analysis is complete, you can check the results in IC Compiler by displaying maps or error cells. If any issues are found, you can debug and fix the issues in the design. The ability to check the results and fix the design issues without leaving the IC Compiler session greatly reduces the overall design cycle time.

Note:

IC Compiler supports only PrimeRail static analysis in version C-2009.06.

If you encounter any issues during IC Compiler rail analysis, debug the issues by first looking at the information in the analyze_rail.log file. You might also require the stand-alone version of PrimeRail for some detailed debugging. When the debugging is complete in PrimeRail, you can then export the results and load them back into IC Compiler where you can check the problematic areas and fix them in the design.

This section includes the following topics:

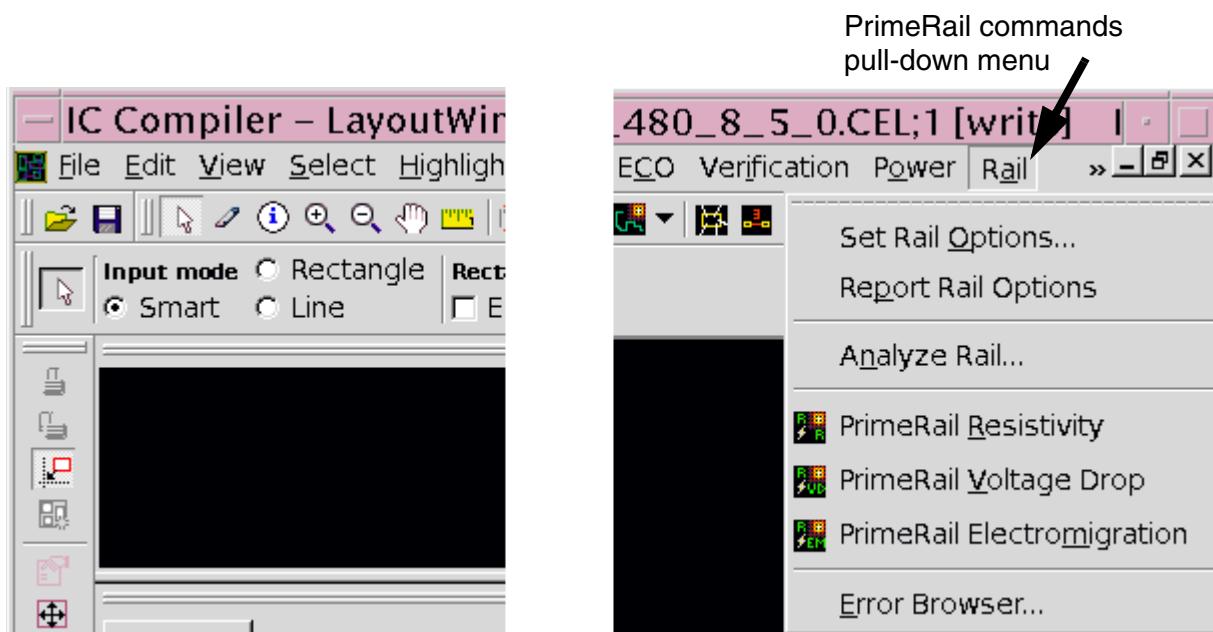
- [Executing PrimeRail Commands in IC Compiler](#)
- [Specifying Setup Options for Rail Analysis](#)

- [Reporting Rail Option Settings](#)
- [Performing Rail Analysis](#)
- [Reading and Removing Map Data Files](#)

Executing PrimeRail Commands in IC Compiler

You can execute PrimeRail commands on the `icc_shell` command line. If you invoke the IC Compiler graphical user interface by executing `gui_start` in `icc_shell`, all the PrimeRail commands reside under the RAIL category in the IC Compiler GUI, as shown in [Figure 7-28](#).

Figure 7-28 Executing PrimeRail Commands from IC Compiler Graphical User Interface



The following IC Compiler commands are used to complete the PrimeRail tasks.

- `set_rail_options`
- `report_rail_options`
- `analyze_rail`

Specifying Setup Options for Rail Analysis

You can specify setup options for performing targeted rail analysis on specified nets. If you do not run this command, the `analyze_rail` command is run using the default options.

To specify the setup options for rail analysis,

1. Choose Rail > Set Rail Options.

The Set Rail Options dialog box appears.

Alternatively, you can use the `set_rail_options` command.

2. Set the following options, depending on your requirements:

- Use pins as pads – If you select this option, the top-level PG pins are treated as ideal voltage sources in the block-level simulation. The default is enabled.
- Pad master file – When the design to be analyzed is the top level of a chip and the pad cells are placed and connected to the power and ground network, you can specify the name of the pad master file, which specifies the Milkyway pad master cells. The pad master cells will be used as ideal voltage sources for rail analysis.
- Pad instance file – When your design has pad cells hooked up to your network of choice but you want to set only some of the Milkyway pad cell instances to be used as the ideal voltage sources for rail analysis, you can specify the name of the pad instance file.
- Tap file – If you are working with a block or a design that does not yet have pins defined, or perhaps the location or design of the pad cells is not finalized, you can specify a user-defined tap file that defines the physical locations (coordinates and layer numbers) of the ideal voltage sources during rail analysis by using the following format:

```
netName layerNumber xCoord yCoord
```

The coordinates are in microns. For instance, if you want to apply the ideal voltage of the VDD net on layer 3 at three locations, create the file with the following lines:

```
VDD 3 784.000 1826.000  
VDD 3 1184.000 1826.000  
VDD 3 1584.000 1826.000
```

IC Compiler user-defined tap points which are described in a power network analysis format and which have a defined layer that sits on top of a PG geometry, are also accepted.

If a user-defined tap file is provided to PrimeRail in the existing text box in the PG Rail Analysis dialog box by using the `poRailAnalysis` command, the file can contain either PrimeRail or power network analysis syntax and it will be interpreted accordingly. The mixing of PrimeRail and power network analysis syntax is not allowed and will result in an error message.

- Package file – The package file includes general linear packaging models during full-chip rail analysis. You can specify the name of an optional SPICE file containing packaging parasitics to be analyzed together with the chip.

The tool supports package circuits with mutual inductors (K), dependent sources, like voltage-controlled voltage sources (E), current-controlled current sources (F), voltage-controlled current sources (G), current-controlled voltage sources (H) and sub-circuit instances (X) for packaging in the rail analysis.

- Voltage drop – Specify a voltage drop threshold value in millivolts for violation checking. If you do not specify a value, the tool does not generate a voltage drop violation report
- Switching Activity – Specify an optional switching activity file in a value change dump (VCD) or switching activity interchange format (SAIF) to be used in rail analysis. The default is VCD. For VCD activity files, the rail analysis is performed across all time values included in the file.

These files are often generated by simulation tools that use a different top-level design than the IC Compiler session. In this case, you can use the Strip Path text box to enter a portion of the object names in the top-level that are not represented in the IC Compiler session and should be removed.

- Full chip analysis script files – Specify a SDC script file or parasitic SPEF file for full chip analysis. You can also specify an optional Verilog netlist file or UPF script file. If these files are not specified, they are created, based on the data saved in the Milkyway database. The full chip SDC and signal parasitics are required.
- Executable directory – Specify the path to the PrimeRail or PrimeTime binary. By default, IC Compiler locates the binary by using the UNIX search path
- Output directory – Specify the name of the output directory for the `analyze_rail` command. By default, the command creates a directory called `/pr_current_cell_name` in the working directory.
- Hosts – Specify a computer host name if you want to run PrimeRail on a different machine. By default, it runs on the same machine on which the IC Compiler session is running.

3. Select OK or Apply.

Reporting Rail Option Settings

You can report the current rail option settings for the `analyze_rail` command by choosing Rail > Report Rail Options.

Alternatively, you can use the `report_rail_options` command.

Performing Rail Analysis

You can use the `analyze_rail` command to perform rail analysis and to check on the specified PG nets. When executed, the command generates the data that is need to run PrimeRail and runs a PrimeRail script in batch mode within the IC Compiler session. This

command supports power and ground network integrity analysis, voltage drop analysis, and electromigration analysis. When the analysis is finished, a PrimeRail command script file is generated. If errors occur, you can modify the file and load it back into the tool in a subsequent run. For debugging purposes, you can load the generated script file in the stand-alone version of PrimeRail and complete the rest of the analysis.

For more information on debugging rail analysis in the stand-alone version of PrimeRail, see [“Debugging IC Compiler—PrimeRail Flow” on page 7-133](#).

To perform rail analysis on the specified nets,

1. Choose Rail > Analyze Rail.

The Analyze Rail dialog box appears.

Alternatively, you can use the `analyze_rail` command.

2. Set the following options, depending on your requirements.

- PG nets – Specify the names of the PG nets on which to perform the targeted rail analyses.
- PG network weakness analysis – Enables connectivity checking on power and ground network of the current design. Select this option when the design being implemented in IC Compiler has finished global routing, clock tree synthesis, detailed routing, or chip finishing.

PG network integrity analysis checks for missing vias, insufficient vias, pin connections, and other related PG network issues. It also calculates the minimum path resistivity on the target PG nets and saves the results in a map data file, which can be viewed in the resistivity map.

- Voltage drop analysis – Enables static voltage drop analysis to make sure a voltage drop threshold is met. Select this option when the design being implemented in IC Compiler has finished clock tree synthesis, detailed routing, or chip finishing, and you have executed the `analyze_rail -integrity` command. Recommendations are provided at the end of the analysis for fixing the design.

Voltage drop analysis analyzes the voltage drop of the given PG nets and saves the results in a map data file, which can be viewed in the voltage drop map.

- Electromigration analysis – Enables electromigration analysis on power nets to make sure the current densities are under electromigration thresholds. Select this option when the design being implemented in IC Compiler has finished clock tree synthesis, routing or chip finishing, and you have executed `analyze_rail -integrity` and `analysis_rail -voltage_drop`.

Electromigration analysis calculates the current density or pin currents and saves the results in a map data file, which can be viewed in the electromigration map.

When the analysis is completed, the command reports the analysis results in the log file. If any electromigration violations are found, recommendations are provided for resolving the problems.

- PrimeRail script generation analysis only – Generates and writes out a PrimeRail script for the target analysis (integrity checking, voltage drop, electromigration) without running PrimeRail. This allows you to modify and reuse the script file as desired.

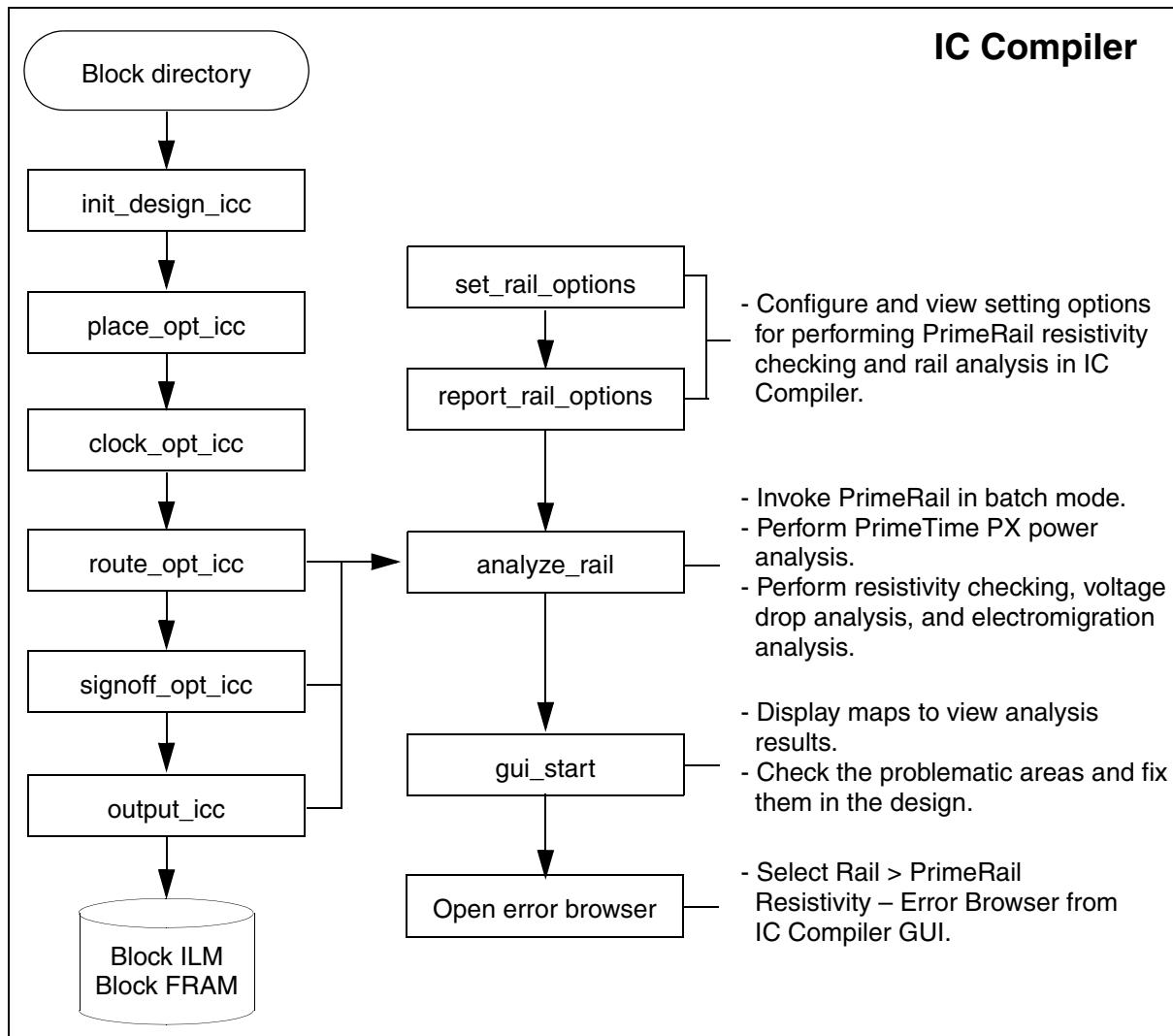
Note:

For information on map data files, see “[Reading and Removing Map Data Files](#)” on [page 7-132](#).

3. Select OK or Apply.

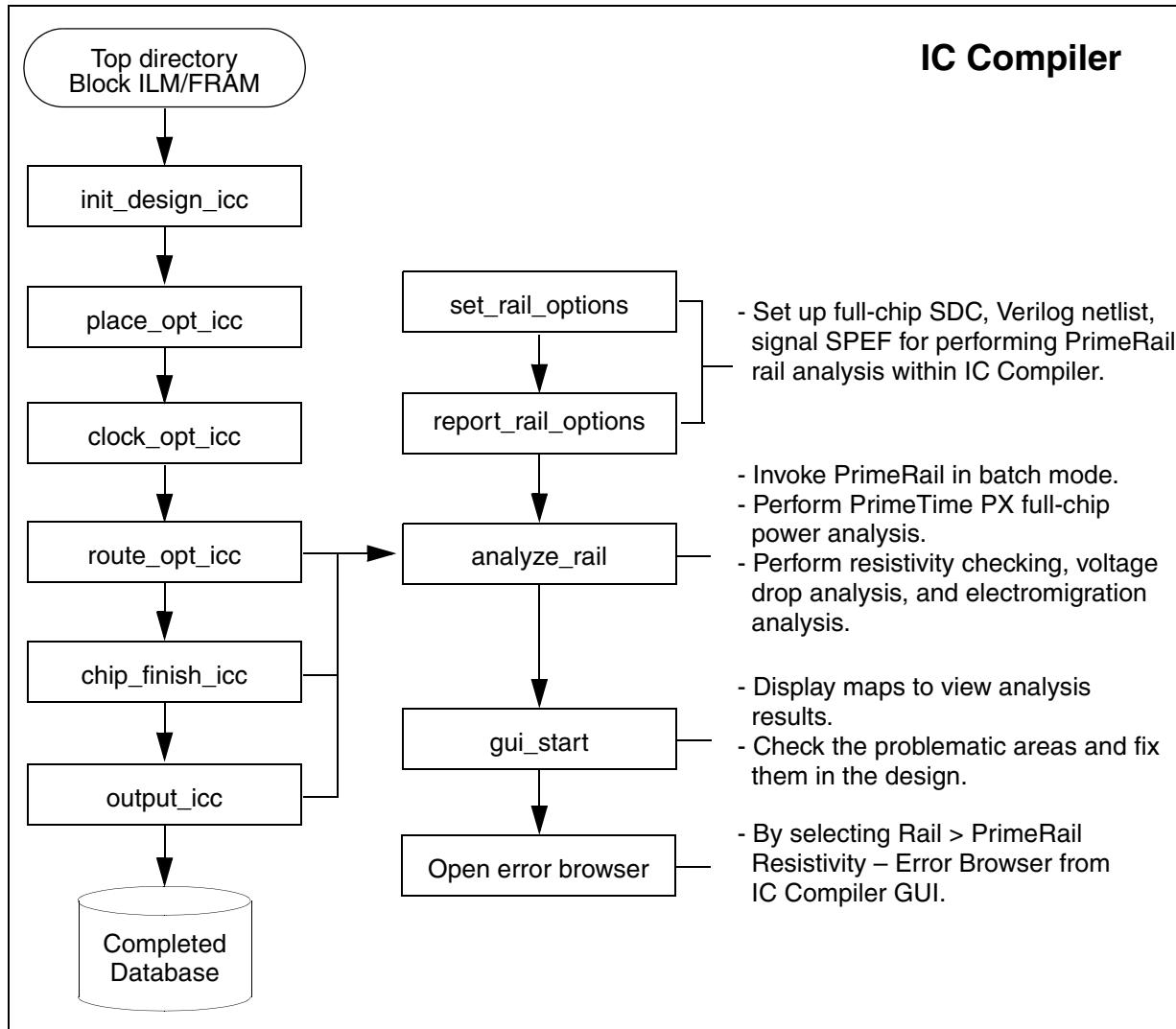
[Figure 7-29 on page 7-131](#) describes the commands used in block-level rail analysis within the IC Compiler flow.

Figure 7-29 PrimeRail Block-Level Rail Analysis Command Flow Within IC Compiler



[Figure 7-30 on page 7-132](#) describes the commands used in chip-level rail analysis within the IC Compiler flow.

Figure 7-30 PrimeRail Chip-Level Rail Analysis Command Flow Within IC Compiler



Reading and Removing Map Data Files

Use the `read_rail_maps` command to read rail map data from a map file generated by PrimeRail. The command loads the map data file into IC Compiler to enable the display of PrimeRail resistivity, voltage drop, or electromigration maps that are produced during rail analysis. You must specify the name of the map data file to load into IC Compiler.

Use the `remove_rail_maps` command to remove all rail map data from the current IC Compiler session.

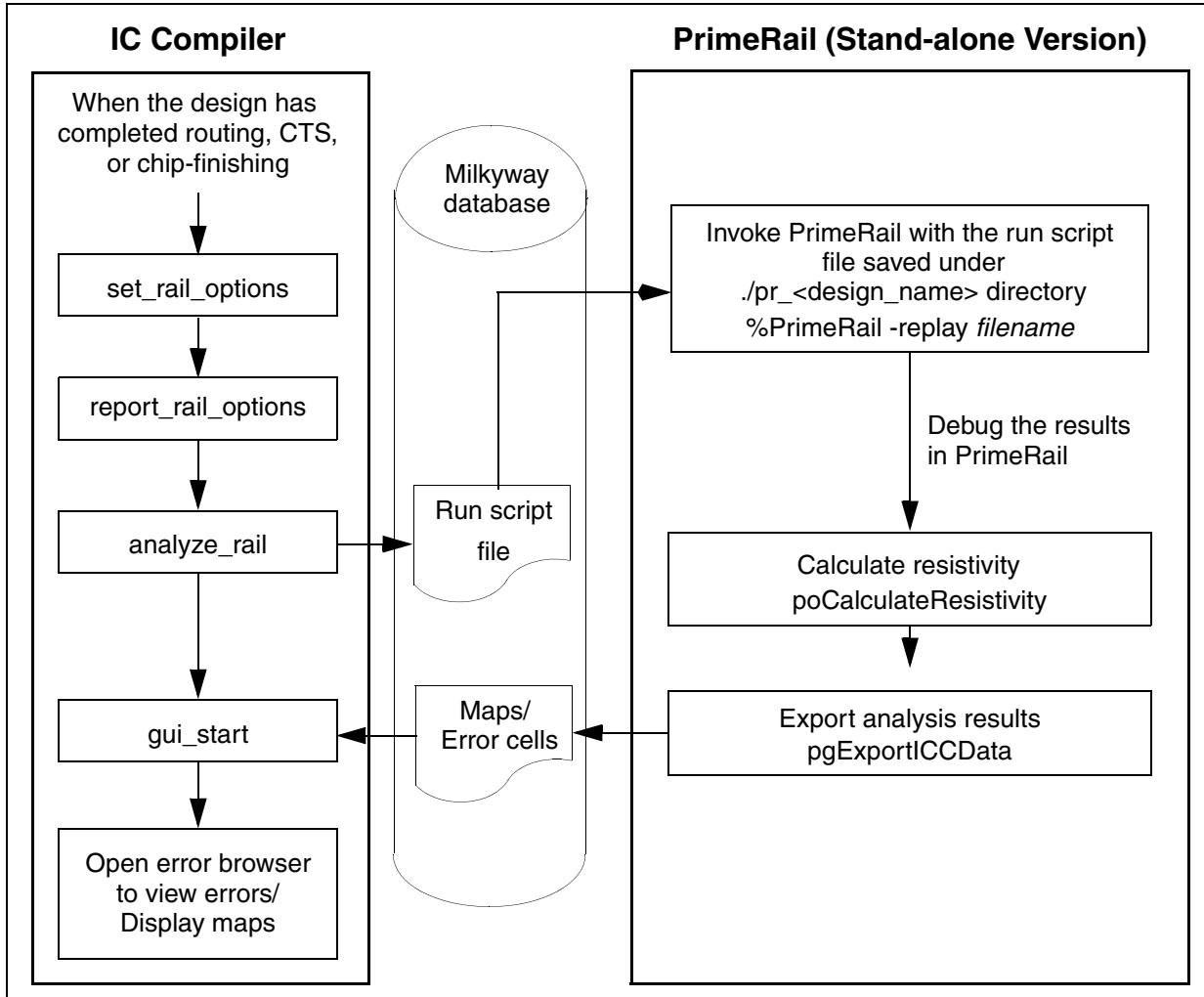
Debugging IC Compiler—PrimeRail Flow

If errors occur during IC Compiler rail analysis, you can locate the run script file from the `./pr_design_name/` directory under your working directory, where IC Compiler saves all the output and log data related to PrimeRail rail analysis. Note that the run script file and log file are rewritten each time the `analyze_rail` command is executed. If you want to keep the script and log files for debugging purposes, be sure to save a copy of the files on your disk.

Invoke the stand-alone version of PrimeRail by using the run script file that IC Compiler generates and by debugging any reliability issues in PrimeRail. You can then export the analysis results to an output file and display the results in IC Compiler for map display and error browsing.

[Figure 7-31 on page 7-134](#) illustrates the steps for debugging in the IC Compiler and PrimeRail analysis flow.

Figure 7-31 Debugging in the PrimeRail and IC Compiler Flow



Calculating Net Resistivity

When power and ground parasitic extraction is complete using the `poExtractPGParasitics` command, execute the `poCalculateResistivity` command if you want to calculate resistivity values from the nets in the design. When the calculation is complete, run the `pgExportICCData` command to export the calculated resistivity results to an output file for map display in IC Compiler. The calculated resistivity results are saved to the RAIL view.

Note:

If you plan to view the resistivity map through the `pgMap` command in the stand-alone version of PrimeRail, you do not need to execute the `poCalculateResistivity` command.

The `poCalculateResistivity` command uses the following syntax:

```
poCalculateResistivity '("netName1" "netName2") "userDefineTapFileName" \
    useTopLevelDesignPin "pinResistanceFileName" \
    "padFileType=Master|Instance" "padFileNameName"
```

Argument	Description
<code>userDefineTapFileName</code>	The name of the tap file that specifies where the ideal voltage is applied by coordinate and layer number.
<code>useTopLevelDesignPin</code>	Specify in order to use the top-level design pin. The default setting is 1. Otherwise, set the argument to 0.
<code>pinResistanceFileName</code>	The name of the pin resistance file.
<code>padFileType</code>	Specify the pad file type to be Master or Instance.
<code>padFileName</code>	The name of the pad file.

You need to provide at least one argument so PrimeRail can locate boundary conditions for resistivity calculation. Other STRING type arguments can be an empty string " ".

Example

If you want to calculate resistivity from the nets VDD, VDDV, and VDDX, and the boundary condition is to use top-level design pins, execute the `poCalculateResistivity` command using the following syntax:

```
poCalculateResistivity '("VDD" "VDDV" "VDDX") "" 1 "" "" "" "
```

Exporting PrimeRail Analysis Data for IC Compiler

If you finish the rail analysis tasks in the stand-alone version of PrimeTime and want to view the results in IC Compiler, execute the `pgExportICCDATA` command to write the calculated analysis data that was saved in the RAIL view to an output file for IC Compiler resistivity, voltage drop or electromigration map display. Error reports and error cells generated from voltage drop and electromigration analyses will also be generated if needed.

To export analysis data for IC Compiler,

1. Enter pgExportICCData. The Export ICC Data dialog box appears.
2. Select “Combined analysis” if you are performing a full-chip analysis.
3. Select the name of the power or ground net whose RAIL view data you want to generate.
4. In the “Output directory” text box, enter the name of the directory where you want to save the output data and error files to be generated.
5. In the "Maps to write" section, choose the type of map data to be generated.
6. In the "Error cell/report" section, select "Write error cell" and specify the name to be assigned to the error cell. The default naming convention for error cells is *cell_name_net_name_export.err*. You can open and view the generated error cells in the IC Compiler error browser.

To generate an error report file, select "Write error report" and specify the file name you want to assign to the error report to be generated. The default name convention for error reports is *cell_name_net_name_vdlem.txt*.

Select either “EM errors” or “Voltage Drop Errors,” making your choice based on whether you want to generate the results of electromigration analysis or generate the results of voltage drop analysis.

Sample Script

The following is a sample script to export electromigration analysis data for displaying PrimeRail analysis results in IC Compiler.

```
pgExportICCData
setFormField "Export ICC Data" "use combined analysis result" "0"
setFormField "Export ICC Data" "P/G Net Names" "VDD"
setFormField "Export ICC Data" "directory to write maps and error file to" "rail_map_data"
setFormField "Export ICC Data" "Include Resistivity map" "1"
setFormField "Export ICC Data" "include VD map" "1"
setFormField "Export ICC Data" "include EM map" "1"
setFormField "Export ICC Data" "Write an error cell" "1"
setFormField "Export ICC Data" "Write an error report" "1"
setFormField "Export ICC Data" "Report EM errors" "1"
formOK "Export ICC Data"
```

Using the `create_rail_setup` Command to Perform Reliability Analysis

You can also use the `create_rail_setup` command to prepare the data required by PrimeRail for performing reliability analysis. By default, this command creates a directory called `synopsys_rail_setup` in the current working directory and outputs the files necessary to run rail analysis to this directory. You can change the name of the output directory by using the `-directory` option.

When you run the `create_rail_setup` command on a block (the default), the following files are output:

- The PrimeRail setup file (`synopsys_pr_setup.e`)

This binary file defines the search path, link path, TLUPlus files, operating conditions, Milkyway design library, Milkyway cell, parasitic corner, and power and ground nets.

- The Verilog netlist (`design.v`)

By default, IC Compiler generates a Verilog netlist for the current Milkyway design. To use an existing Verilog netlist, specify the file name by using the `-verilog` option.

- The SDC constraints for the design (`design.sdc`)

By default, IC Compiler generates an SDC file for the current Milkyway design. To use an existing SDC file, specify the file name by using the `-sdc` option.

- The signal parasitics (`design.spf.max` or `design.spec.min`)

By default, IC Compiler performs RC extraction and generates a SPEF file for the maximum corner of the current Milkyway design. If RC extraction has already been done, use the `-no_rc_extract` option to skip the RC extraction. To generate a SPEF file for the minimum corner instead of the maximum corner, use the `-parasitic_corner min` option. To use an existing SPEF file, specify the file name by using the `-spef` option.

To generate files for the top level in a hierarchical flow, use the `-hierarchy` option when you run the `create_rail_setup` command. When you use the `-hierarchy` option, only the PrimeRail setup file and the Verilog netlist are generated. You must provide the signal parasitics generated by Star-RCXT and the SDC file.

By default, the `create_rail_setup` command saves the design in a cell called `design_pr`. If you do not want to save the design, use the `-no_save` option.

8

Performing Prototype Global Routing

You can perform prototype global routing to get an estimate of the routability and congestion of your design. Global routing is done to detect possible congestion “hot spots” that might exist in your floorplan due to the placement of the hard macros or inadequate channel spacing.

This chapter includes the following sections:

- [Setting Up for Routing](#)
- [Running Global Prototype Routing](#)

Setting Up for Routing

You can specify general routing setups for IC Compiler to use whenever you perform routing.

This section includes the following topics:

- [Setting Routing Options](#)
 - [Setting Route Guides](#)
 - [Setting the Preferred Routing Direction](#)
 - [Specifying Net Layer Constraints](#)
 - [Setting Nondefault Routing Rules](#)
 - [Setting Routing Types](#)
 - [Setting Net Aggressors](#)
-

Setting Routing Options

You can specify global routing, track assignment, detail routing, and other routing options for IC Compiler to use whenever you perform routing functions.

To specify routing options,

1. Choose Route > Routing Setup > Set Route Options.

The Set Route Options dialog box appears, displaying the global routing options.

Alternatively, you can use the `set_route_options` command.

2. Click the tab (Global Routing, Track Assign, Detail Routing, or Miscellaneous) for the type of options you need to specify.

3. Specify your routing requirements.

If you need to specify options for another routing stage, repeat steps 2 and 3.

4. Click OK.

To report your settings, use the `report_route_options` command.

Setting Route Guides

You can create or remove route guides that do the following for a specific area of your design:

- Prevent routing for signal or prerouted nets
- Change the wiring direction
- Control wiring density
- Fix violations

To create a route guide,

1. Choose Floorplan > Create Route Guide.

The Create Route Guide dialog box appears.

Alternatively, you can use the `create_route_guide` command.

2. Specify the route guide characteristics.

- Specify the route guide by typing its name in the Name box.
- To repair violations, select the “Repair as single SBox” check box.
- To prevent the routing of signal wires on specific layers, select the “No signal route on layers” check box and select the layers in the list below the check box.
- To prevent the routing of wires within a certain distance of the route guide where there is a keepout margin, do not select the “Zero min-spacing” check box.
- To prevent automatic preroutes on layers, select the “No automatic preroutes on layers” check box and select the layers in the list below the check box.
- To specify the allowable horizontal and vertical track utilizations, select the “Route track utilization” check box and specify the horizontal and vertical track utilization percentages.
- To switch the preferred wiring direction, select the “Switch preferred direction” check box.
- Specify the coordinates for the route guide by typing the coordinates in the dialog box or drawing the rectangle in the layout view.

Also, specify that the route guide should snap to the minimum grid, placement site, routing track (the default), middle routing track, or user grid.

3. Click OK or Apply.

You can find route guides by using a filter expression or rectangular areas that encompass or touch the guides.

To find route guides,

- Use the `get_route_guides` command with the appropriate options.

To do this	Use this option
Find route guides that meet the conditions of an expression	<code>-filter expression</code>
Find route guides that are totally within a rectangular area	<code>-within x1 y1 x2 y2</code>
Find route guides that touch or cross over the border of a rectangular area	<code>-touch x1 y1 x2 y2</code>

For more information on the `get_route_guides` command, see its man page.

To remove route guides,

- Use the `remove_route_guides` command with the appropriate options.

To do this	Use this option
Remove all route guides	<code>-all</code>
Remove specific route guides by name	<code>-name name</code>
Remove all route guides that belong to a collection	<code>collection</code>

Setting the Preferred Routing Direction

Use the `set_preferred_routing_direction` command to reset and override the default preferred routing direction specified in the library or the design for a specific layer. The layer direction set with this command applies to the current design only.

The `set_preferred_routing_direction` command uses the following syntax:

```
set_preferred_routing_direction
    -layers list_of_layers
    -direction horizontal | vertical
```

For example, to set the preferred routing direction to vertical for layer M5 and M7, enter

```
icc_shell> set_preferred_routing_direction -layers "M5 M7" \
    -direction vertical
```

Note:

Settings with the `create_route_guide -switch_preferred_direction` command, which changes the preferred direction within the area that is covered by the route guide, will override the settings made with the `set_preferred_routing_direction` command.

Use the `report_preferred_direction` command to report the preferred routing direction for all the routing layers. The report lists the library and user-defined routing directions, as well as the direction that the tool will use.

Here is a sample report:

```
*****
Report : Layers
Design : core_chip
Version: Y-2006.06
Date   : Thu Mar 23 03:43:06 2006
*****
Layer Name      Library      Design      Tool
understands
metal1          Horizontal    Not Set    Horizontal
metal2          Vertical     Not Set    Vertical
metal3          Horizontal    Not Set    Horizontal
metal4          Vertical     Not Set    Vertical
metal5          Horizontal    Vertical   Vertical
metal6          Vertical     Vertical   Vertical
```

Use the `remove_preferred_routing_direction` command to remove the user-defined directions for a specific layer from the design. The syntax is

```
remove_preferred_routing_direction -layers list_of_layers
```

Specifying Net Layer Constraints

IC Compiler lets you specify which layers can be ignored for routing and which layers are to be ignored for RC and congestion estimation. If you do not specify layers to ignore for RC and congestion estimation, those functions use the layers you specify for routing. You can specify these layers in the following ways:

- Specify a minimum routing layer

Sets that layer and the layers above it for routing. If you don't specify layers to ignore for RC and congestion estimation, those functions use all of the routing layers.

- Specify a maximum routing layer

Sets that layer and all layers below it for routing. If you don't specify layers to ignore for RC and congestion estimation, those functions use all of the routing layers.

- Specify minimum and maximum routing layers

Sets the layers you specify and all layers between them for routing. If you don't specify layers to ignore for RC estimation and congestion, those functions use all of the routing layers.

- Specify the layers to ignore for RC and congestion estimation

To specify these layers, use the `set_ignored_layers` command with its options.

To do this	Use this
Specify the lowest routing layer	<code>-min_routing_layer layer_name</code>
Specify the highest routing layer	<code>-max_routing_layer layer_name</code>
Specify the lowest and highest routing layers	<code>-min_routing_layer min_layer_name -max_routing_layer max_layer_name</code>
Specify the layers to ignore for RC and congestion estimation	<code>-rc_congestion_ignored_layers layer_name...</code>

For example, to define layers M2 through M7 for routing, use the `set_ignored_layers` command as follows:

```
icc_shell> set_ignored_layers -min_routing_layer M2 -max_routing_layer M7
```

To define the same layers for routing and to force the RC and congestion estimation functions to ignore the M1, M2, M8, and M9 layers, use the following command:

```
icc_shell> set_ignored_layers \  
-min_routing_layer M2 -max_routing_layer M7 \  
-rc_congestion_ignored_layers {M1 M2 M8 M9}
```

Setting Nondefault Routing Rules

You can define nondefault or default routing rules for specific nets. These rules, which define wire width and spacing rules and via types, are defined by the `define_routing_rule` command.

To set a net routing rule,

- Define the routing rule by following these steps:

- Choose Route > Routing Setup > Define Routing Rule.

The Define Routing Rule dialog box opens.

Alternatively, you can use the `define_routing_rule` and `set_net_routing_rule` commands.

- Type the rule name in the “New rule name” box.
- If you are using a nondefault reference rule, select the Specified option and enter the reference rule name.
- Specify a number for the taper level.
- To snap the shielding wires to the track, select the “Snap to track” check box.
- To use multipliers for the layer width and spacing, specify the layer width multiplier in the “Multiplier width” box and specify the layer spacing multiplier in the “Multiplier spacing” box.
- In the Metal table, specify the width, spacing, shield width, shield spacing, and whether to use in the rule for each layer.
- In the Contact table, specify the contact, row, column, and whether to use in the rule for each set of layers.
- Click OK.

2. Apply the routing rule to nets by following these steps:

- Choose Route > Routing Setup > Set Net Routing Rule.
The Set Net Routing Rule dialog box appears.
- Specify the nets that the rule applies to, in the Nets box.
- Specify the name of the nondefault routing rule in the “Routing rule” box.
- As needed, specify the top layer probe mode, net reroutability, and timing-driven spacing.
- Click OK.

Setting Routing Types

For debugging, you can set or change the routing types of wires, vias, or paths to indicate which IC Compiler routing engine is to route them.

To specify a routing of any type,

1. Choose Route > Routing Setup > Set Route Type.

The Set Route Type dialog box appears.

Alternatively, you can use the `set_route_type` command.

2. Type the nets or a collection of nets in the “Wire/Via/Path objects” box.
3. Specify the net type as follows:
 - For signal nets, select “Detail route” or User.
 - For clock nets, select Ring, Strap, “Tie off detail route,” or User.
 - For power or ground nets, select Ring, Strap, “Tie off detail route,” User, “Standard cell pin connection,” or “Macro I/O pin connection.”
4. Click OK or Apply.

To remove a routing type,

1. Choose Route > Delete Route.

The Delete Route dialog box appears.

Alternatively, you can use the `remove_route_by_type` command.

2. Specify the nets to remove by doing the following:
 - To remove types of nets, select the types.
 - To remove specific nets, browse for or type their names in the “Net names” box.
3. Click OK or Apply.

Setting Net Aggressors

To minimize crosstalk-induced noise, you can specify the aggressors for a victim net. Crosstalk prevention during postroute optimization uses that information to minimize the crosstalk-induced noise from the aggressor nets.

To specify aggressors for a victim net,

1. Choose Route > Routing Setup > Set Net Aggressors.

The Set Net Aggressor dialog box appears.

Alternatively, you can use the `set_net_aggressors` command.

2. Specify the victim net in the “Victim net” box.
3. Specify the aggressor nets in the “Aggressor nets” box.
4. Click OK or Apply.

Running Global Prototype Routing

During global routing, IC Compiler assigns nets to the global routing cells through which they pass. For each global routing cell, the routing capacity is calculated according to the blockages, pins, and routing tracks inside the cell. Although the nets are not assigned to the actual wire tracks during global routing, the number of nets assigned to each global routing cell is noted. IC Compiler calculates the demand for wire tracks in each global routing cell and reports the overflows, which are the number of wire tracks still needed after IC Compiler assigns nets to the available wire tracks in a global routing cell.

IC Compiler might reduce overflows by detouring nets around congested areas and increasing the wire length. You can examine the global routing report that appears in the command window and display congestion maps to help you decide whether your design can be routed.

The global router considers spacing and wide-wire variable routing rules as well as shielding variable routing rules, when calculating congestion.

Global routing is divided into the following phases:

- One initial routing phase, where all the unconnected nets are routed
- One or more rip-up and reroute phases, where for a selected set of nets, the routing results from the previous phase are deleted and nets are rerouted to reduce the congestion

Note:

If, after the second routing phase, the maximum overflow in any direction is greater than 50, both global routing and prototype routing stop because the design is too congested and it is now unroutable.

To perform global prototype routing do the following:

1. Choose Route > Prototype Global Route.

The Proto Route dialog box appears.

Alternatively, you can use the `route_fp_proto` command.

2. Set the options, depending on your requirements.

Next to Effort, select one of the following:

- “Low” (the default) runs initial global routing only. (It is recommended for fast prototyping feedback.)
- “Medium” runs initial global routing and two rerouting passes. (It is recommended for detailed floorplan analysis.)

- “Congestion map only” specifies that a congestion map be displayed.
One-dimensional and two-dimensional congestion maps that indicate the routing congestion in the chip are displayed.
During data preparation, the core area is divided into global route cells (GRC) based on the average height of the standard cells. During global routing, the tool calculates routing congestion according to the availability of wire tracks inside the global route cells.
- “Track assignment” performs track assignment after global routing.
Track assignment specifies which tracks within each global routing cell are to be used for each net. Track assignment operates on the entire design at once; it can make long routes straight and reduce the number of vias.
After track assignment finishes, all nets are routed but not very carefully. There may be many violations, particularly where the routing connects to pins. Run detail routing (`route_detail` command) to correct the violations.

3. Click OK or Apply.

9

Performing Clock Planning

This chapter describes how to reduce timing closure iterations by performing clock planning on a top-level design during the early stages of the virtual flat flow, after plan groups are created and before the hierarchy is committed. You can perform clock planning on a specified clock net or on all clock nets in your design.

For hierarchical timing closure, clock planning can also be used to generate realistic clock latency through timing budgeting to fix timing violations.

Clock planning tries to minimize clock skew by running block-level and top-level clock tree synthesis during the early stages of the design flow to determine the clock budgets, allocate resources for clock buffers and clock routes, determine optimal clock pin locations for soft macros, and provide an estimate of the block-level insertion delays and skew for each plan group prior to finalizing the floorplan. Having optimal clock pin locations is a key factor in meeting the final clock skew and insertion delay numbers with the optimal number of buffers.

This chapter includes the following sections:

- [Setting Clock Planning Options](#)
- [Performing Clock Planning Operations](#)
- [Generating Clock Network and Source Latency for Each Clock Pin of Each Plan Group](#)
- [Using Multivoltage Designs in Clock Planning](#)
- [Performing Plan Group-Aware Clock Tree Synthesis in Clock Planning](#)

- [Supporting Abutted Floorplans in Hierarchical Clock Planning](#)

Setting Clock Planning Options

Before you can compile clock trees inside the plan groups and build clock trees at the top level, you must first set different clock planning options such as anchor cell insertion, specifying nets for clock planning, and whether or not to route the clock nets after clock planning. You can also modify the clock tree constraint settings.

To set clock planning options,

1. Choose Clock > Set Clock Plan Options.

The Set Clock Plan Options dialog box opens.

Alternatively, you can use the `set_fp_clock_plan_options` command.

2. Set the options, depending on your requirements.

- Clock Nets – Enter the name of the clock nets on which to do clock planning.
- No Feedthroughs in Plan Groups – Enter a list of plan groups on which you do not want buffers placed. During the top-level clock tree synthesis phase of clock tree planning, no buffers are placed on the plan groups, unless they drive sinks inside those plan groups. This minimizes the creation of feedthroughs.
- Anchor Cell – Enter the name of the cell that is inserted as an anchor cell for all the plan groups. The anchor cell is a clock buffer cell. All plan groups should use the same buffer type. Inverters are not supported. This option is required. If you do not specify the name of the anchor cell, the IC Compiler tool issues an error message.

For each clock interface net (nets that cross plan group boundaries), an anchor cell (driver cell) is inserted for each plan group on every input clock net that crosses a hierarchical block. This partitions the plan group level clock subtree from the top level clock tree.

An isolation cell is also inserted at the top level (for each output clock port on the plan group) to isolate the plan group level clock subtree from the top level clock tree.

The anchor cell is inserted inside the plan group at the center of the mass of flip-flops that are connected to each clock net to isolate the top-level clock net from the clock net that is inside the plan group. The input pin of the anchor cell is driven by the clock net, and the output pin of the anchor cell drives the root clock pin of the block.

- Route Mode – Choose whether or not to route the top level clock nets after clock planning. Based on the routing information, clock pins are created and assigned a location where the route crosses the plan group boundary.

Global route – Select this option to perform global routing on the clock nets.

Detailed route – Select this option to perform detail routing on the clock nets.

None – Select this option if you do not want to route the clock nets. This is the default.

- Output directory – Enter the name of the directory in which to write out all constraint files, log files, and generated reports from clock planning. The default directory is /cp_output.
- Keep block level tree – Select this option if you want to keep the top-level clock tree buffers inside the plan groups during clock planning. The default is to remove the block level clock tree when clock planning is complete.

Clock tree planning might, for example, show a large insertion delay and skew value on one of the blocks in your design. By keeping the clock tree buffers inside the plan groups, you can more easily analyze them to determine why clock tree planning shows such a large value.

- Set clock tree options – Select this option to open the Set Clock Tree Options dialog box.

3. Click OK or Apply to set the clock planning options.

Reporting Clock Planning Options

You can get a report of the clock plan options by using the `report_fp_clock_plan_options` command. If no clock plan options have been set, this command reports the default values.

Removing Clock Planning Options

You can remove (reset) the database entries for the clock planning options you set by using the `reset_fp_clock_plan_options` command.

Performing Clock Planning Operations

Clock planning is done during the early stages of the virtual flat flow (after plan groups have been created and before the hierarchy is committed) to allocate clock resources and provide an estimate of the block-level insertion delay and skew for each plan group, prior to finalizing the floorplan.

You can perform clock planning operations to compile the clock trees inside plan groups and build clock trees at the top level based on the options you have selected in the Set Clock Plan Options dialog box (`set_fp_clock_plan_options` command).

To perform clock planning operations,

1. Choose Clock > Compile Clock Plan.

The Compile Clock Plan dialog box appears.

Alternatively, you can use the `compile_fp_clock_plan` command.

2. Set the options, depending on your requirements.

- Operation Condition – Select the operating conditions (Max, Min, or Min/Max) for top-level clock tree synthesis and optimization. The default is Max.
- Insert Anchor only – If you select this option, the clock planning tool only inserts anchor cells on the input ports of the plan groups, and does not synthesize the clock plan.

By default, the clock planning tool inserts anchor cells on the input ports of the plan groups plan groups, and then synthesizes the clock plan.

3. Select OK or Apply.

During clock planning, the following operations are performed:

- Anchor cells are inserted on the input ports of the plan groups to isolate the clock trees inside the plan groups from the top-level clock tree.
- Fast clock tree synthesis is run inside each plan group to estimate the insertion delay and skew values at the input of the anchor cells.
- The clock tree synthesis results are annotated on floating pins, which are defined on the anchor cells.
- The top-level clock tree is synthesized to the anchor cell floating pins.
- Detail routing is run on the clock interface nets.

Generating Clock Tree Reports

You can use clock skew analysis to generate a skew report for a specified clock (or for all the clocks within a design) before or after routing. You can view the report in a text window or write it to a specified file.

To generate clock tree reports, choose Clock > Report Clock Tree or use the `report_clock_tree` command. By default, the global skew is reported for all the generated clock trees.

Generating Clock Network and Source Latency for Each Clock Pin of Each Plan Group

For hierarchical timing closure, clock planning is also used to generate realistic clock latency through budgeting to fix timing violations. Top-level timing violations result not only from delays on combinational logic between registers, but also from clock skew between launching and capturing registers.

When you perform clock planning operations, it enables the timing budgeter to generate clock network and clock source latencies for each clock pin of each plan group in the design. (See [“Performing Clock Latency Budgeting” on page 12-15](#).)

- Clock network latency is the time a clock signal (rise or fall) takes to propagate from the clock definition point in the design to a register clock pin.

Example:

```
set_clock_latency 2 -max -rise [get_clock CLK]
```

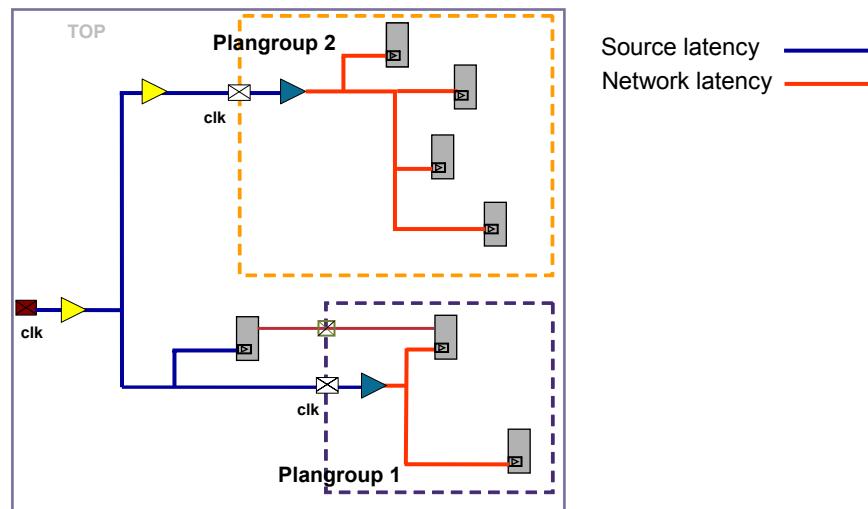
- Clock source latency is the time a clock signal takes to propagate from its ideal waveform origin point to the clock definition point in the design.

Example:

```
set_clock_latency 2 -source -max -rise [get_clock CLK]
```

[Figure 9-1](#) shows an example of the clock network latency and the clock source latency.

Figure 9-1 Clock Network and Source Latency



Creating a Virtual Clock for I/O Paths

For each plan group, a virtual clock can be created to describe clock latency outside of the plan group

The format for defining virtual clocks created for clock latencies outside the plan group is

- Clocks launching flops of input paths:

clock_name_v_in

- Clocks capturing flops of output paths:

clock_name_v_out

Note:

To minimize the number of virtual clocks, only the worst-case clock latency is created for all input and output pins of plan groups launched or captured by a virtual clock.

[Figure 9-2 on page 9-7](#) shows a virtual clock example.

- In plan group 1, the name of the input path is in1.

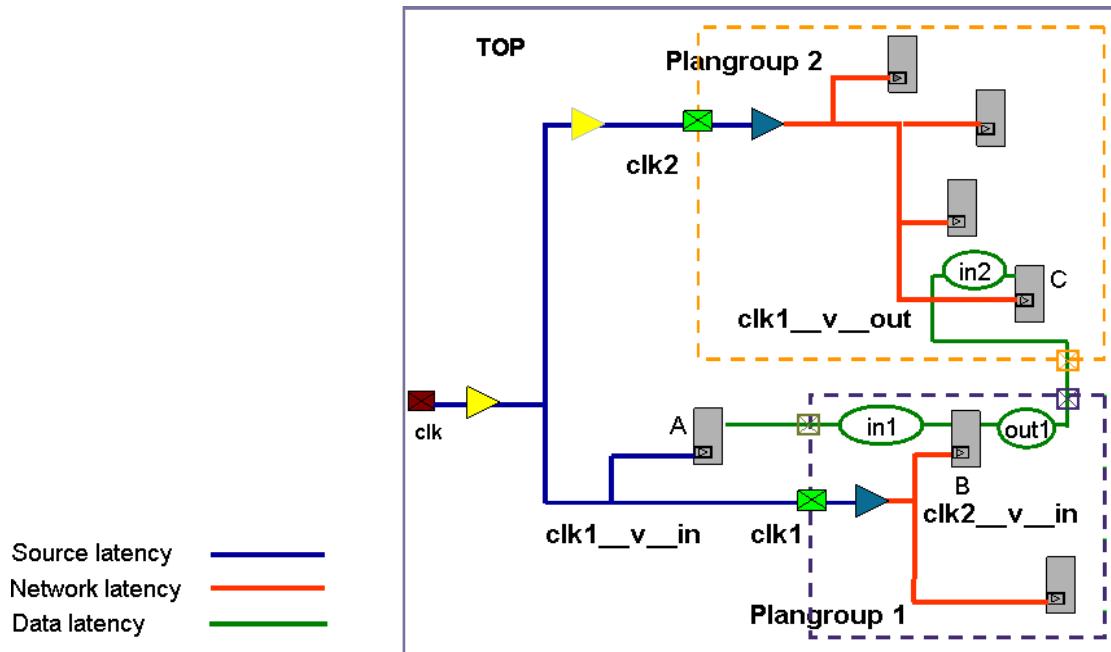
clk1_v_in launching flop A on the top level has source latencies only.

- In plan group 1, the name of the output path is out1.

clk1_v_out (*clk2*) capturing flop C in plan group 2 has both source and network latencies.

- In plan group 2, the name of the input path is in2.

clk2_v_in (*clk1*) launching flop B in plan group 1 has both source and network latencies.

Figure 9-2 Virtual Clock Example

Using Multivoltage Designs in Clock Planning

Clock planning supports multivoltage designs. Designs in multivoltage domains operate at various voltages. Multivoltage domains are connected through level-shifter cells. A level-shifter cell is a special cell that can carry signals across different voltage areas.

Clock planning isolates each plan-group-level clock from the top-level clocks by inserting an anchor cell. However, if you are using multivoltage designs in your clock planning, level-shifter cells should have already been inserted into the voltage area of the plan group.

Interpreting Level-Shifter Cells as Anchor Cells During Clock Planning

If there is a level-shifter cell for an interface clock net (a net that comes from the top level clock net into a plan group), it is interpreted as an anchor cell. Then, during clock planning, anchor cells are inserted for each interface clock net only as long as they do not cross voltage areas. This prevents the insertion of buffers and inverters into the wrong voltage areas. Both plan-group-level clocks and voltage areas now isolated from the top-level clocks.

Note:

Feedthroughs created by clock planning may not honor the voltage area constraints. For all designs in multivoltage domains, you should specify the “No Feedthroughs in Plan Groups” option on the Set Clock Plan Options dialog box (`set_fp_clock_plan_options` command).

Performing Plan Group-Aware Clock Tree Synthesis in Clock Planning

You can perform plan group-aware clock tree synthesis in clock planning. With this feature, clock tree synthesis can:

- Generate a clock tree that honors the plan groups while inserting buffers into the logic hierarchy tree or, if it exists, into the corresponding physical region. The physical region can be a voltage area, an exclusive move bound, or a plan group.
- Prevent new clock buffers from being placed on top of a plan group unless they drive the entire subtree inside that particular plan group. This results in a minimum of clock feedthroughs, which makes the design easier to manage during partitioning and budgeting.

To perform plan group-aware clock tree synthesis in clock planning with fully abutted floorplans, set the following variable:

```
set cp_full_abut_cts_region_aware true
```

To control plan group-aware clock tree synthesis in clock planning, click in the text box next to the “No Feedthroughs in Plan Groups” option on the Set Clock Plan Options dialog box and enter a list of plan groups where clock feedthroughs should not be generated.

Supporting Abutted Floorplans in Hierarchical Clock Planning

You can perform hierarchical clock planning (`compile_fp_clock_plan` command) on designs with fully abutted floorplans. This can help solve possible DRC violations on nets going through different plan groups.

To do this, clock tree synthesis must be plan group-aware (See “[Performing Plan Group-Aware Clock Tree Synthesis in Clock Planning](#)”).

From a placed design, set the following variable to perform hierarchical clock planning on fully abutted floorplans:

```
set cp_in_full_abut_mode true
```

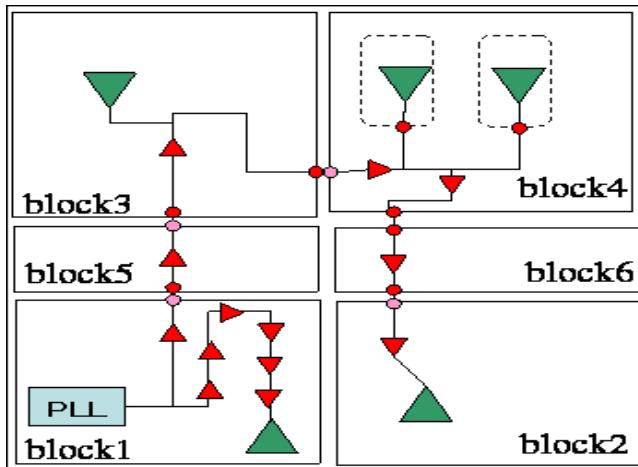
Run the `compile_fp_clock_plan` command to compile the clock trees inside plan groups and build clock trees at the top level.

Note:

Clock tree planning should resolve any DRC violations on nets going through different plan groups in the design, and the ideal clock buffer location is far from the plan group that is associated with the logical hierarchy to which the buffer is added.

[Figure 9-3](#) shows an example of a fully abutted floorplan.

Figure 9-3 Fully Abutted Floorplan



10

Performing In-Place Timing Optimization

In-place timing optimization is used to improve the timing of a given design, and in particular, to meet the timing constraints on the design. It is an iterative process based on virtual routing. You can run in-place optimization before or after global routing. If you decide to run in-place optimization after you have run global routing, the timer will use the routes to provide the timing, but the optimization is still based on the virtual routes.

This chapter includes the following sections:

- [Running In-Place Timing Optimization](#)
- [Running Trace Mode In-Place Optimization](#)
- [Using In-Place Optimization With Multivoltage Designs](#)
- [Performing In-Place Optimizations Based on Pin Locations](#)

Three types of optimizations are performed: timing improvement, area recovery, and fixing timing design rule violations. These optimizations preserve the netlist's logical hierarchy, and the physical locations of most cells change as little as possible.

The optimization process starts with the most significant changes on multiple violation paths that will quickly improve overall timing. Next, the scope is narrowed to focus on individual paths. Netlist changes are committed only if the timing improves. The output is a legally placed netlist that is plan group aware.

Many methods for improving timing are used, including

- Inserting buffers and inverters (legal locations are found automatically)
In the case of inverters, it is guaranteed that the polarity of the signals is preserved
- Increasing or decreasing cell size (If the cell size is increased, the larger cell is automatically adjusted to ensure that all cells are placed in legal locations.)
- Moving cells

Similar to cell sizing, the cells are placed in legal locations

Running In-Place Timing Optimization

To perform in-place timing optimization,

1. Choose Timing > In Place Optimization.

The In Place Optimization dialog box appears.

Alternatively, you can use the `optimize_fp_timing` command.

2. Set the options, depending on your requirements.

- Effort

You can specify how much effort is used to minimize the worst negative slack in the design. If you select an effort level of high, more effort is expended to improve the timing of the design, resulting in more CPU time. In-place optimization stops when it finds the worst negative slack in the design cannot be further improved. The output is a legally placed netlist.

The default effort is medium.

- Fix design rule

Enable this option to fix design rules. The default is disabled. Design rule violations, such as maximum transition, maximum capacitance, and maximum fanout, are fixed by use of buffer insertion, gate sizing, and automatic high-fanout net (HFN) synthesis for handling medium- and high-fanout nets.

Note:

The default high-fanout net synthesis threshold is 100. To change this threshold, use the `set_ahfs_options` command.

When you run in-place optimization at the virtual route stage, only obvious design rule violations are fixed because wire locations and interconnect are estimated at this stage since timing analysis cannot be completely accurate and runtime is shorter. After you finish global routing, optimization takes longer to run, but the results are based on more accurate timing information.

- **Area recovery**

Enable this option to direct the in-place optimization engine to invoke additional operations to try to reduce the cell area without causing timing violations. The area is optimized by removing cells and decreasing the size of the cells on noncritical timing paths. This allows more space for optimization on critical paths.

The default is disabled.

Note:

Area recovery is an operation that tries to reduce the total area of cells used in the design. Area recovery will not cause timing to become worse on the critical paths, but some paths might see an increase in timing as long as it is nonviolating. If the path has positive slack, area recovery might reduce this slack to zero. Area recovery does not cause nonviolating paths (paths with non-negative slack) to become violating paths.

Area recovery, however, is a difficult and expensive operation, resulting in longer runtime but with a smaller area being used. Designs with high utilization will benefit from area recovery but with a cost that results in higher runtimes.

- **Only add buffers for feedthrough nets**

Enable this option to only add buffers for feedthrough nets. Optimization is not performed and timing is not updated. Using this option minimizes design changes in the late stages of design planning. The default is disabled.

You must run the `analyze_fp_routing` command before you use this option; otherwise, the tool issues an error message and stops.

The tool automatically selects medium-sized buffers and inserts them near each pin (port) location determined by the `analyze_fp_routing` command. During the optimization phases, in-place optimization might resize or move these buffers. The buffers are added in the same hierarchy as the plan groups.

- **Report quality of results**

Enable this option to get reports on the worst negative slack (WNS) and total negative slack (TNS) of the design, the utilization percentage of each plan group and top level, the number of buffers added, and the number of cells sized for each plan group and top level.

The default is disabled.

3. Click OK or Apply.

Running Trace Mode In-Place Optimization

You use the `optimize_fp_timing` command or choose Timing > In-Place Optimization, in-place optimization is performed on the full chip netlist in virtual flat mode.

Another technique for improving capacity and runtime is to run in-place optimization in trace mode. In this mode, you can optimize all top-level paths or only interface timing paths by using the `set_fp_trace_mode` command to mark a design and then load a subset of the design into trace mode. This mode is useful since most designers are concerned about top level and interblock timing at the virtual flat placement stage.

In trace mode, IC Compiler only loads the data structures associated with the marked design objects (netlist information) are loaded into memory. These are the interface timing paths and the top-level netlist. This means that the data structures will not be complete (sparse netlist); the complete netlist exists only in the Milkyway database.

The syntax is

```
set_fp_trace_mode  
[-verbose]
```

Note:

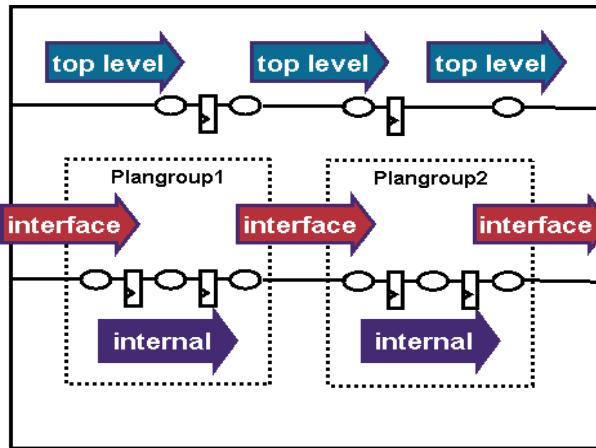
To determine if a design is already in trace mode, use the `get_fp_trace_mode` command.

When trace mode timing is used in a flat design with plan groups defined for soft macros, trace mode timing selectively filters out paths by tracing only the nets and cells of timing paths at the top level, which cross plan group boundaries. Accordingly, the timer (`report_clock_timing` command) reports only the timing violations on those paths. By using trace mode timing analysis, you can detect and fix these timing violations in the early phases of the design flow. This enables faster timing analysis by focusing on interblock nets.

During the in-place optimization stage, only the interface nets between the top level and blocks will be optimized, which results in faster runtime and higher capacity.

[Figure 10-1 on page 10-5](#) illustrates the top-level, internal, and interface paths for trace mode timing.

Figure 10-1 Top-level, Interface, and Internal Paths for Trace Mode Timing



Removing the Trace Mode

To remove trace mode timing and return to the normal mode for timing analysis, use the `end_fp_trace_mode` command. This command will remove the trace mode design from memory. Once the next command that accesses the netlist of the design is used, it will trigger a reload of the entire design netlist.

Using Timing and Data Query Commands in Trace Mode In-Place Optimization

You can use the following commands during trace mode:

- Timing commands

```
check_timing
get_timing_paths
report_clock_timing
report_disable_timing
report_timing
report_timing_derate
report_qor
extract_rc -estimate
```

- Data query commands

```
get_attribute
get_cells
get_clocks
get_nets
```

```
get_object_name  
get_pins  
get_ports  
get_selection
```

Note:

Commands not on this list do not operate on the trace mode netlist and generate an error message.

Using In-Place Optimization With Multivoltage Designs

The `optimize_fp_timing` command detects multivoltage settings and automatically turns on this feature during in-place optimization. By following predefined multivoltage design rules, in-place optimization is able to improve timing by replacing a cell instance with another cell of a different size from within the same voltage domain by moving cells within the same voltage domain and by inserting buffers without violating multivoltage design rules.

Performing In-Place Optimizations Based on Pin Locations

You can call the `optimize_fp_timing` command after pin cutting (finalize routing). Timing optimization understands pin locations decided by the previous pin cutting command when performing timing optimization. It chooses medium-sized buffers and inserts them near each pin (port) location determined by the `analyze_fp_routing -finalize_pins_feedthroughs` command.

11

Performing Routing-Based Pin Assignment

You can create top-level soft macro pins in hierarchical designs, constrain the pins during pin creation, edit and modify the soft macro pins, and analyze the quality of the pin assignment results.

IC Compiler provides two ways to perform pin assignment: on soft macros, the traditional pin assignment, or on plan groups, using the pin-cutting flow.

This chapter includes the following sections:

- [Setting Pin Assignment Constraints](#)
- [Setting Voltage Area Feedthrough Constraints](#)
- [Performing Traditional Pin Assignment](#)
- [Aligning Soft Macro Pins](#)
- [Removing Soft Macro Pin Overlaps](#)
- [Performing Pin Assignment on Plan Groups \(Pin-Cutting Flow\)](#)
- [Using Pin-Cutting With Multiply Instantiated Modules](#)
- [Performing Incremental Pin-Cutting](#)
- [Creating Rectangular or Rectilinear Pin Guides](#)
- [Performing Pin and Feedthrough Analysis](#)

- [Removing Feedthrough Ports and Nets From Blocks and Voltage Areas](#)
- [Refining the Pin Assignment](#)

Setting Pin Assignment Constraints

You can set constraints prior to performing pin assignment. These pin constraints are honored during pin assignment on soft macros during traditional pin assignment and on plan groups in the pin-cutting flow. The pin assignment constraints are saved in the Milkyway database.

This section includes the following topics:

- [Specifying Physical Design Constraints on Pins](#)
- [Reporting Pin Assignment Constraints](#)

Various pin constraint settings are available and can be applied to all soft macros or to a selected set of soft macros. Pin creation layers can be limited to a set of selected layers. Pins can be constrained to avoid corner placement by various factors. You can also set the spacing between pins and preroutes.

To assign pin constraints,

1. Choose Pin Assignment > Pin Constraints.

The Pin Assignment Constraints dialog box appears.

Alternatively, you can use the `set_fp_pin_constraints` command.

You can apply constraints to all soft macros and plan groups or to specified soft macros or plan groups. By default, the specified constraints apply to all soft macros and plan groups.

You can apply constraints to block-level pins instead of soft macro pins. To do this, enable the “Block level pins” option. The default is disabled.

- Specified Ports – Enter a list of specified ports to which the specified block-level non-edge pin constraints apply.

2. Click the Add button to open the Add Pin Assignment Constraints dialog box.

3. Under “Pin Constraints”, click the Settings tab and complete the necessary constraint settings.

Allowed layers from – Select this option to specify the range of consecutive metal layers on which to place soft macro pins.

- Specify the range of allowed layers, from the lowest to highest allowed layers.

Retain existing pins – Select this option when you do not want the pin assignment to delete movable pins. The default is off.

Pin Stacking – Specifies how to handle pin overlaps across layers. You can select the following options:

- No Stacking – Select this option when you do not want to overlap pins on two different layers (for example, metal1 pins with metal3 pins). This is the default.
- Stacking OK – Select this option if you want signal, and power and ground pins to be stacked.
- Allow signal Stacking – Select this option if you want only signal pins to be stacked.
- Allow signal-power/ground Stacking – Select this option if you want both signal, and power and ground pins to be stacked

Hard constraint for pin – Select this option if you want the pin-to-pin spacing constraint or location to be treated as a hard or soft constraint. Pin assignment always tries to honor the constraints you set. However, if a design is very congested and there are not enough wire tracks to allocate all the pins, the pin-to-pin spacing constraint may be violated. To prevent this violation, select this option.

The default is off and the pin spacing constraint is treated as a soft constraint.

- Spacing – If you select this option, the pin-to-pin spacing constraint is treated as a hard constraint. If there are insufficient pin slots to honor the constraint, pin assignment issues an error message.
- Location – If you select this option, the pin location constraint is treated as a hard constraint. Pins are created at the exact locations. If there are pin spacing violations, pin assignment issues a warning message.

Pin spacing – Pins that are created are always snapped to wire tracks. Specify the number of wire tracks in the spaces between adjacent pins. The default pin-to-pin spacing is 1 extra wire track between pins.

Pin preroute spacing – Specify the number of wire tracks between adjacent preroutes and pins. The default distance between preroutes and pins is 3 wire tracks.

To allow enough space for routing, no pins are created on the section of a soft macro edge that is parallel to a preroute if that edge is closer to the preroute than to the number of wire tracks you specify. Pins can be created on other sections of the soft macro edge if the distance from that section of the soft macro edge is at or at least equal to the pin-to-pin spacing.

Prevent corner pins by – Specify the distance from the corners of soft macros where pins should begin. You can specify the distance two ways:

- Number of wire tracks – Specify the corner keepout spacing as the number of wire tracks from the corner where the pin assignment should not create pins. The default is 5 wire tracks.

The pin assignment engine takes the number of wire tracks from the corners specified in the dialog box, multiplies it by the maximum metal layer pitch (the maximum wire track distance for metal layers from minimum to maximum), and then uses the resulting distance as the corner spacing. For example, if the minimum layer is metal3 and the maximum layer is metal6, and metal6 pitch is the largest, the corner spacing is calculated as metal6 pitch multiplied by the number of wire tracks from the corners of soft macros.

- Percentage of Macro side length% – Specify the corner keepout spacing as a percentage of the edge length where the pin assignment should not create pins. The default is 5.0 percent.

Keep bus bits together – Select this option if you want pins associated with bus bits to be grouped together. The default is off.

Order bus bits – During automatic ordering of bus pins, IC Compiler honors all the bits of a bus as one individual entity and follows the ordering of bits when assigning pins. All pins of a bus are grouped together on a single edge. Bit ordering can be done in one of the following ways:

- From LSB to MSB – Pins are ordered from the least significant bit to the most significant bit and from left to right on a horizontal edge and bottom to top on a vertical edge.

For example:

`a[0], a[1], a[2], ... a[31]` for the 32-bit bus "a"

- From MSB to LSB – Pins are ordered from the most significant bit to the least significant bit and from left to right on a horizontal edge and bottom to top on a vertical edge.

For example:

`a[31], a[30], ... a[1], a[0]` for the 32-bit bus "a"

- Scrambled – Bits are first sorted in increasing order and then ordered in the following fashion:

```
skip count = 1
a[0], a[2], a[4],...a[30], a[1], a[3], a[5],...a[31]
skip count = 2
a[0], a[3], a[6],...a[1], a[4], a[7],... a[2], a[2],
a[5], a [8],...
```

The default skip count is 1.

- Consistent wire length – Bits are ordered so that they equalize the wire lengths of the bus bits between soft macros.
4. Under “Pin Constraints,” click the Feedthrough Settings tab and complete the necessary feedthrough settings.

Allow Feedthroughs – Select this option when you want pin assignment or pin-cutting to create at least two feedthrough ports in the child cell. Each top-level net for which a feedthrough port is created is split into a set of new top-level nets and child-level nets (if feedthrough nets were created for the child net). Directions are assigned for the newly created feedthrough ports. Because new ports are created in soft macros, the netlist is modified as well. The default is off.

Exclude feedthroughs on clock nets – Select this option to exclude feedthrough ports on clock nets. The default is on.

Exclude scan chain nets – Select this option to exclude feedthrough ports on scan chain nets. The default is on.

Exclude network – Select this option to exclude not only the nets specified by the other exclusion options, but also the nets connected through combinational logic to those specified nets. These nets are also excluded from feedthrough creation during pin assignment. The default is on.

Exclude feedthroughs on nets with fanout – Select this option to exclude feedthroughs generated on nets with fanout greater than the specified number. The default number is 0. You can change the high-fanout threshold in the associated text box. The default is off.

Exclude feedthroughs on specified nets – Select this option to exclude feedthroughs on specified nets. The file should contain top-level nets, and the format is one net per file. The default is off.

Exclude pins on specified soft macro sides – Select the side of the macro where you do not want pins created. Use the left, right, top, or bottom for rectangular macros.

Enter a side number for rectilinear macros.

5. Click Apply to write the pin constraint settings to the CEL views of the selected set of soft macros.

Specifying Physical Design Constraints on Pins

You can use the `set_pin_physical_constraints` command to specify the physical design constraints on pins. The constraints are stored in the Milkyway database.

The `set_pin_physical_constraints` command uses the following syntax:

```
set_pin_physical_constraints
[-pin_name pin_name]
[-cell cell_name]
```

```

[-layer layers]
[-width]
[-depth]
[-offset]
[-order]
[-off_edge center | location | auto]
[-location]
[object]

```

The `set_pin_physical_constraints` command uses the following arguments to set the physical constraints per pin instance.

<code>pin_name</code>	Name of the pin on which the physical constraint is applied Valid values: Name of any pin in the design
<code>cell</code>	Name of the Milkyway cell to which the pin belongs. This option also requires the pin name.
<code>layer</code>	Name or number of the metal layers on which you want to place the pin. Only a single metal layer is supported. Valid values: Name or number of any layer defined in the technology file or -1 to indicate that any layer can be used
<code>width</code>	Width of the pin in microns Valid values: Any positive floating-point number or 0 (the default) to indicate the minimum possible width If the width constraint specified is smaller than the minimum width defined in the technology file, then the minimum width is applied.
<code>depth</code>	Depth of the pin in microns Valid values: Any positive floating-point number or 0 (the default) to indicate the minimum possible depth If the depth constraint specified is smaller than the minimum width defined in the technology file, then the minimum depth is applied.
<code>side</code>	Side of the block on which the pin is placed so that it abuts the specified side of the block. The side number is a positive integer. Given any rectangular or rectilinear shape, the lower-left edge is the starting edge, which is side 1. Moving in a clockwise direction, the side of the next edge is 2, and so forth. A value of 0 indicates no side constraint.

<code>offset</code>	Relative offset distance in microns of the pin placement from the edge of the cell boundary. The default is 0. Valid values: Any positive floating-point number
<code>order</code>	Order of the pin placement relative to the position of other pins on the same side Valid values: Any positive integer Pins with lower order values are placed closer to the bottom or left edge of the boundary than pins with higher values. Otherwise, the pin order values have the following effects:
	<ul style="list-style-type: none"> • If a pin has a value of 0, the placement of the pin relative to the other pins is not important. This is the default. • If a pin has a value of 1, no pin can be placed between the pin and the corner of the boundary. • If a pin has a value of 2, any number of pins can be placed between the pin and the corner of the boundary. • If two pins have consecutive values, no pin can be placed between them. • If a group of pins has the same value, no other pins can be placed between any two of the pins in the group. In this case, the ordering inside the group is not important.
<code>off_edge center location auto</code>	Determines the type of off-edge pin. By default, the command sets on-edge pin constraints. The following supported values are mutually exclusive. <p><code>center</code> – Places the pin at the center of the cell</p> <p><code>location</code> – Places the pin at exact coordinates of the specified location point. The pin is marked as fixed.</p> <p><code>auto</code> – Places the pin in an optimal position guided by a user-specified pin guide</p>
<code>location</code>	Specifies the coordinates of the off-edge pins. You must supply the <code>off_edge location</code> option.
<code>object</code>	Specifies the pin to which the specified pin constraints apply. The command accepts only one object.

Honoring the Pin Physical Constraints

If you want pin assignment to honor any preexisting pin physical constraints that are set by the `set_pin_physical_constraints` command or the `read_io_constraints` command, use the `set_fp_pin_constraints -use_physical_constraints on | off` option and set the value to `on`. By default, this option is off so that pin assignment does not use physical constraints. See “[Specifying Physical Design Constraints on Pins](#)” on page 11-5.

Reporting Pin Assignment Constraints

You can use the `report_fp_pin_constraints` to display the pin assignment constraints that you have set for specified soft macros, plan groups, or ports. These constraints control both pin assignment and pin-cutting.

The syntax is:

```
report_fp_pin_constraints  
[objects]  
[-block_level]
```

Use the `objects` argument to include a collection of blocks or ports.

Note:

A mix of blocks and ports is currently not allowed.

Blocks specify the soft macros and plan groups for which the pin constraints are shown. Ports specify the ports for which the block level non-edge pin constraints are shown. When you specify the ports, you must also specify the `-block_level` option.

Use the `-block_level` option to report the block-level pin assignment constraints on the currently open cell instead of the pin constraints for blocks contained in the cell.

Setting Voltage Area Feedthrough Constraints

You can set voltage area feedthrough constraints that are honored during both global routing and finalize routing. The voltage area constraints are saved in the Milkyway database.

To set the voltage area feedthrough constraints,

1. Choose Pin Assignment > Voltage Area Constraints.

The Voltage Area Constraints dialog box appears.

Alternatively, you can use the `set_fp_voltage_area_constraints` command.

2. Select the “All” option if you want the feedthrough constraints to apply to all the voltage areas (the default) in your design, or select the “Specified” option and enter a list of voltage areas to which the specified feedthrough constraints apply.

For example, to create feedthrough nets on voltage areas A, B, and C, select the “Specified” option and enter the following in the text box:

```
[get_voltage_areas A B C]
```

3. Select the “All feedthroughs” option if you want global routing and routing analysis to create feedthrough ports in the logical child cell.

Each top-level net for which a feedthrough port is created is split into a set of new top-level nets and child-level nets if feedthrough nets were created for the child net. Pin directions are assigned for the newly created feedthrough ports.

The default is off; no feedthrough ports are created.

4. Select the “Exclude feedthroughs on specified nets” option if you want to exclude feedthroughs on specified nets. The file should contain top-level nets, and the format is one net per file. The default is off.
5. Click the Set button in the dialog box to apply the feedthrough constraints on the voltage areas.

Reporting Voltage Area Constraints

Use the `report_fp_voltage_area_constraints` command to display the options used to control feedthrough creation on the voltage areas in your design.

Removing Voltage Area Constraints

Use the `remove_fp_voltage_area_constraints` to reset the feedthrough constraints for the specified voltage areas to their default values.

Performing Traditional Pin Assignment

Note:

The pin assignment constraints must be set prior to running pin assignment. (See “[Setting Pin Assignment Constraints](#)” on page 11-2.)

Traditional pin assignment automatically assigns top-level or block-level pins on both rectangular and rectilinear soft macros that mark the points where nets can pass in and out of soft macros.

For top-level pin assignment, the pin assignment engine considers the top-level connections to plan groups, macros, and pad locations when it determines where to assign the pins.

For block-level pin assignment, the pin assignment engine considers the cell placement inside the soft macro when it assigns pins to minimize the wire lengths to the internal connections within the soft macro. Use block-level pin assignment in a bottom-up design flow.

This section includes the following topics:

- [Performing Block Level Pin Assignment](#)
- [Performing Block Level Non-Edge Pin Placement](#)

Note:

The pin assignment results are stored at the child level within the cell master. If you open a child cell after pin assignment and then close it without saving it, the pins are also discarded.

To assign soft macros pins,

1. Choose Pin Assignment > Assign Pins.

The Assign Pins dialog box appears.

Alternatively, you can use the `place_fp_pins` command.

2. Choose whether to assign pins in the current design to all soft macros or to selected soft macros. The default assigns pins to all soft macros.

3. Select the effort used to determine how the pins are placed.

Low – This effort is connectivity driven (flyline-based) and runs very quickly. This is the default.

- For top-level pin assignment, the pin assignment engine considers all the top-level connections for a soft macro pin before choosing the macro side (rectangular or rectilinear) to which the pin is assigned. It also considers the wire track assignments from the routing grid to place pins created for various layers on wire track positions corresponding to those layers.
- For block-level pin assignment, the pin assignment engine considers all the internal connections for a soft macro pin before choosing the macro side (rectangular or rectilinear) to which the pin is assigned.

High – This effort performs a global route. The pin assignment constraints are strictly honored. A high effort generally produces better results.

- For top-level pin assignment, the pin assignment engine performs global routing of the top-level soft macro connections and considers routing congestion as pins are placed.

- For block-level pin assignment, the pin assignment engine performs global routing of the internal soft macro connections to determine the side of the macro on which to assign the pins.

Note:

If feedthrough creation is required (top-level pin assignment only), you must select the high effort.

4. (Optional) Select the Verbose option to control the amount of output messages that are written to the log file during pin assignment.

If the Verbose mode is disabled (off), pin assignment writes out the following messages to a log file during pin assignment:

- Informational messages tracking the progress and stages of the pin creation.
- Informational messages on the success or failure of setting pin assignment constraints on soft macros.
- Error messages regarding issues detected in design data or constraint settings that prevent successful pin creation, such as non-Manhattan soft macro shapes or soft macro sides without enough space to assign all the pins.
- Informational messages on feedthrough generation statistics, such as the number of original ports, and the number of feedthrough ports and feedthrough nets that were created in each soft macro.
- Informational messages on the total number of pins that were created.

If the Verbose mode is enabled (on), in addition to the information it outputs when the Verbose mode is disabled, pin assignment writes the following warning and informational messages to the log file:

- Warning messages regarding issues detected in design data or constraint settings that do not prevent pins from being created, but that might affect the quality of the pin assignment results, such as ports that are not connected, and multidriven or undriven nets.
- Detailed information on feedthroughs, including the names of the feedthrough nets and the ports that are created.
- Detailed information about the created pins, including the name of each created pin and its location.

5. Click OK or Apply to assign pins to the soft macros.

Performing Block Level Pin Assignment

You can assign pin locations based on the current cell placement. Select the “Block level pins” option on the Assign Pins dialog box.

Alternatively, you can use the `place_fp_pins -block_level` command.

Block-level pin assignment is performed separately and independently on each soft macro in the design. The pin assignment engine assigns pins at the child cell level within the soft macro cell, taking into account the cell placement, internal connections, and pin constraints inside the soft macro cell.

Performing Block Level Non-Edge Pin Placement

You can use the `place_fp_pins -block_level` command to place assign pins inside the top-level block directed by the pin guides by using the `create_pin_guide` command and the pin assignment constraints by using the `set_fp_pin_constraints` command.

These non-edge pins are placed within the pin guide region in a location that minimizes the wire length for the particular nets. Specifically, the pins are placed inside the pin guide region at the intersection of the preferred wire track direction on the pin layer with the preferred wire track direction on any layer.

[Figure 11-1](#) and [Figure 11-2](#) on page 11-13 shows examples of region constraints on non-edge pins.

Figure 11-1 Region Constraints on Non-Edge Pins

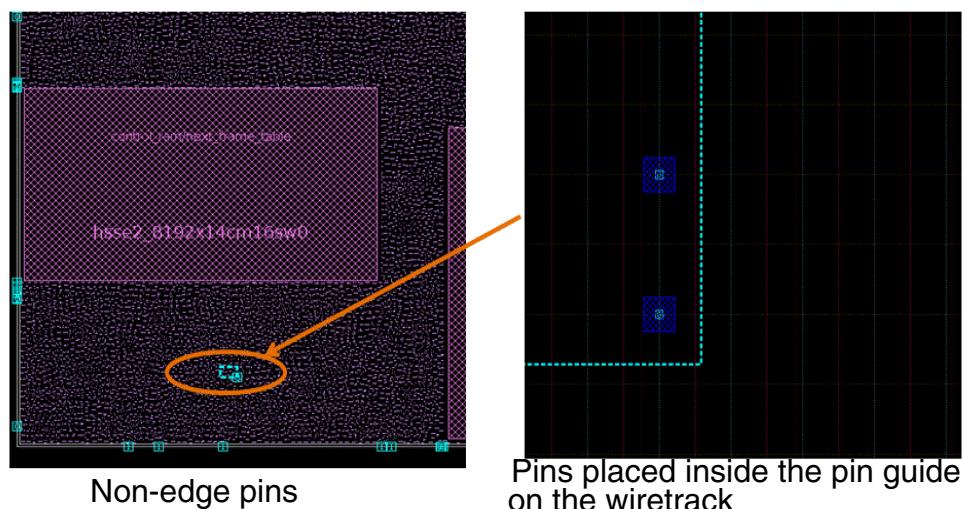
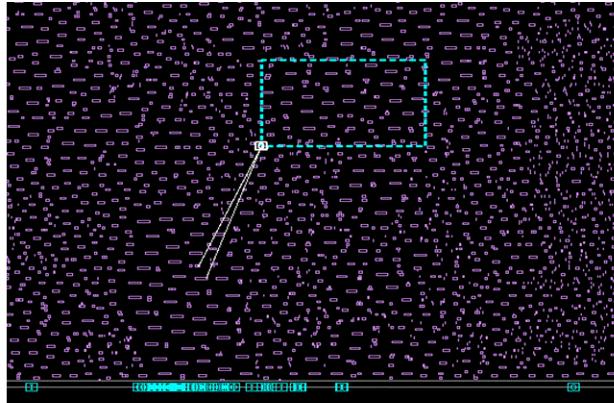


Figure 11-2 Region Constraints on Non-Edge Pins



Pins placed inside the pin guide and close to the standard cell connections to minimize the wire length

Specifying Layers for Non-Edge Pin Placement

Use the `create_pin_guide -allowed_layers layers` command to specify a set of metal layers allowed only for block level non-edge pin placement. If the specified pin guide does not reside entirely within the block, the `-allowed_layers` option cannot be used for pin guides for boundary pins and, therefore, the user-defined pin guide is ignored.

The `layers` argument is a collection of consecutive high metal layers. A collection that contains nonconsecutive metal layers is not supported.

For information on creating pin guides, see “[Creating Rectangular or Rectilinear Pin Guides](#)” on page 11-33.

Checking the Non-Edge Pin Placement

You can check for pins that are not any edges by running the `check_fp_pin_assignment -off_edge` command, and you can check for pins that are not inside the associated pin guide by running the `check_fp_pin_assignment -outside_pin_guide` command.

Aligning Soft Macro Pins

You can align a set of soft macro pins relative to the pins on a reference macro.

To align soft macro pins,

1. Choose Pin Assignment > Align Soft Macro Pins.

The Align Soft Macro Pins dialog box appears.

Alternatively, you can use the `align_fp_pins` command.

2. Click in the Reference Macro text box and enter the cell names of the macros whose pins are to be used as references for alignment.
3. Select the direction in which you want to align the edges of the soft macro pins. This makes a difference only if the reference pins and the pins you want to align have different widths.
 - Left – Aligns the left edges of the soft macro pins. This is the default.
 - Right – Aligns the right edges of the soft macro pins.
 - Top – Aligns the top edges of the soft macro pins.
 - Bottom – Aligns the bottom edges of the soft macro pins.

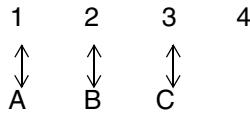
If the pins you want to align sit on the bottom or top edges of their macros and you specify a left direction, the left edges of the pins will be aligned. Conversely, if you specify a right direction, the right edges of the pins will be aligned. In this case, do not specify a bottom or top direction. If you do, the value automatically defaults to left.

If the pins you want to align sit on the left or right edges of their macros, the default value is bottom.

4. Identify the method to use for ordering the reference pins.

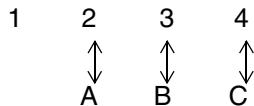
- Low to high – Aligns pairs of soft macro and reference pins in order, starting with pins on the lowest x-coordinate axis for horizontal pins or y-coordinate for vertical pins. This is the default.

If the alignment is done on a horizontal edge, as shown in the following example, and there are four reference pins and three pins on the other macro to be aligned, the fourth reference pin, which is the reference pin with the largest horizontal coordinate, is not used.

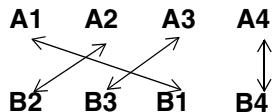


- High to low – Aligns pairs of soft macro and reference pins in order, starting with pins on the highest x-coordinate axis for horizontal pins or y-coordinate for vertical pins.

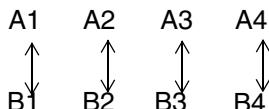
If the alignment is done on a horizontal edge, as shown in the following example, and there are four reference pins and three pins on the other macro to be aligned, the first reference pin, which is the reference pin with the smallest horizontal coordinate, is not used.



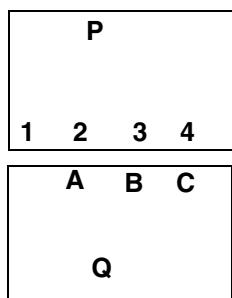
- Net connection – Aligns the soft macro pins to the reference pins if they are connected to the same net, as shown in the following example. A1 and B1 are connected, A2 and B2 are connected, A3 and B3 are connected, and A4 and B4 are connected.



After executing the “Net connection” option, the soft macro pins are aligned as follows. (Suppose macro A is the reference soft macro.)



The following example shows the effects of the “Net connection” and the “High to low” options in which selected pins 2, 3, and 4 of soft macro P are aligned with pins A, B, and C of soft macro Q.



5. Select other options as needed.

- Change layer and width to match reference pin – Select this to change the pin metal layers and widths of the reference pins.
- Align with child macro pins – Select this option to align the pins with the child pins on the reference macro.

The reference macro pins are hard macro pins inside the hard macro. When you use this option, you must also select the “Net connection” option to order the pins by their logical net connections.

- **Mark pins as fixed** – Select this option to fix the pins after aligning them. By default, the pins are unfixed, and they can be moved, using the Edit tool, after you align them.

Removing Soft Macro Pin Overlaps

You can remove pin overlaps between soft macro pins or you can remove overlaps for top level or soft macro pins. Pins are spread so that they do not overlap and do not cause spacing violations. The location of fixed pins are not moved. Pins inside placement blockages are considered illegal and are moved.

The new locations of the pins will be on routing tracks. Selective layer blockage is honored.

Note:

If there are two pins with overlapping pin boundaries but on different layers (for example, metal1 and metal3), it is not considered an overlap.

To remove soft macro pin overlaps,

1. Choose Pin Assignment > Remove Pin Overlaps.

The Remove Pin Overlaps dialog box appears.

Alternatively, you can use the `remove_fp_pin_overlaps` command.

2. To remove pin overlaps for soft macro, select the All option to instruct the tool to remove overlaps for all top level and soft macro pins. This is the default.

To remove pin overlaps for specific soft macros, select the Specified option and enter the names of the macro cells.

3. To remove overlaps for pins, select the All option to instruct the tool to remove overlaps for all pins on the specified macro cells. This is the default.

To remove overlaps for specific pins, select the Specified option and enter the names of the pins.

4. Click OK or Apply.

Note:

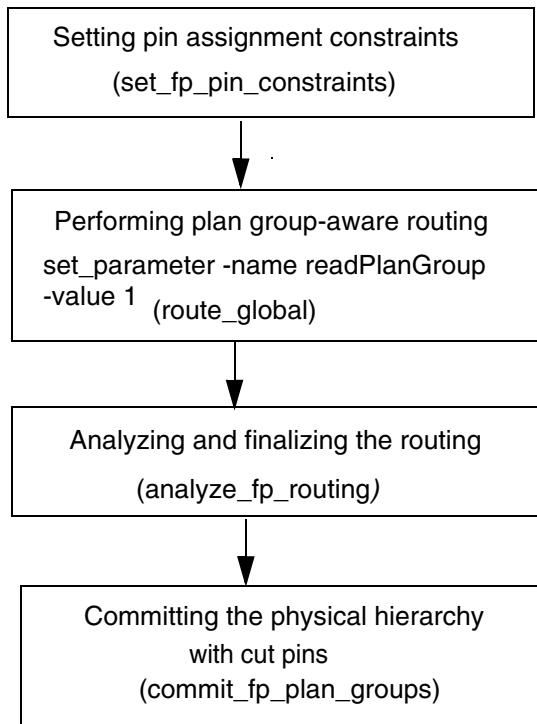
Pin overlap removal might fail if insufficient routing tracks are available or the pin placement is overconstrained. Look for error messages in the console window after performing this operation.

Performing Pin Assignment on Plan Groups (Pin-Cutting Flow)

By using the pin-cutting flow, as shown in [Figure 11-3](#), you can assign pins to plan groups by using the `commit_fp_plan_groups` command. In the pin-cutting flow, the plan group-aware global router generates a routing pattern that is consistent with the future physical hierarchy and with the pin constraints used for assigning pins.

The pin-cutting flow provides you with a much better correlation between the results obtained for physically flat and hierarchical designs after you commit the hierarchy. Pin-cutting allows you to floorplan during placement and fix any floorplan issues early in the design cycle.

Figure 11-3 Pin-Cutting Flow



This section includes the following topics:

- [Setting Pin Assignment Constraints](#)
- [Performing Plan Group-Aware Global Routing](#)
- [Performing Global Routing](#)
- [Analyzing the Pin and Feedthrough Routing](#)

- [Committing the Hierarchy With Cut Pins](#)
-

Setting Pin Assignment Constraints

You must first set your pin assignment constraints (`set_fp_pin_constraints` command) before running plan group-aware global routing. The plan group-aware router will honor the pin assignment constraints for pin-cutting during the commit hierarchy stage.

It is recommended that you exclude feedthroughs for the critical nets of the design, such as

- Clock nets
- Reset, set, and enable signals
- Global DFT signals (for example, test mode and scan enable)
- Scan in and scan out signals

Honoring Top-Down Format Constraints

Specify the `set_fp_pin_constraints` command and the `-use_tdf_constraints` and `-tdf_map_file` options if you want pin assignment to use the top design format (TDF) constraints. The pin-cutting flow honors the TDF physical pin constraints for pin locations on plan groups. These options give you better control over the pin locations for critical nets in the pin-cutting flow.

There are three types of TDF constraints. They are

- Side – The global router places the pin (boundary crossing) on the specified side. The postprocessor maintains the side and keeps the pin, which can be subject to other constraints such as pin-to-pin spacing, at the approximate location specified by the global router.
- Side and Location – The global router obeys the specified side and location within the granularity of one global cell, if possible. The global router also obeys the layer constraint.
- Side and Order – The global router groups the ports on the specified side and places them in order relative to each other, within the granularity of the global cells. Because the order constraint does not affect other ports, other ports of feedthroughs can be placed in the ordered port area. If this is not feasible, it is recommended that you place TDF constraints on all ports on a module.

Precedence and Interaction of TDF and Non-TDF Constraints

The following describes the precedence and interaction of TDF constraints and non-TDF constraints. It is recommended that you do not mix TDF and non-TDF constraints.

- TDF constraints – If there are multiple TDF constraints of the same type for the same pin, the last constraint that is read in takes precedence. For example, if one constraint puts a pin on one side of the soft macro, and later another constraint puts it on the other side, the first constraint is ignored and a warning is issued.
- Non-TDF constraints – In general, any non-TDF constraint, with the exception of bus ordering, takes precedence over TDF constraints. For example, if you specify no ports are to be placed on the left side of a soft macro, but the TDF constraint places a port there, then the TDF constraint is ignored.

Performing Plan Group-Aware Global Routing

You can use plan group-aware global routing to suggest pin locations. To enable plan group-aware global routing, use the `set_fp_flow_strategy` command:

```
set_fp_flow_strategy -plan_group_aware_routing true
```

Note:

If you are running the virtual flat or pin-cutting flow, you must set this value to `true`. If global routing is run without the “awareness” of plan groups, the `analyze_fp_routing` and `commit_fp_plan_groups` commands will not work properly.

By default, plan group-aware routing is set to false (off).

When you set `-plan_group_aware_routing` to `true`, the global router becomes plan group aware. This means that the `route_global`, `route_fp_proto`, and `place_fp_pins` commands will read any plan group definitions and will honor any pin assignment constraints placed upon the plan groups, such as pin spacing and pin blockages.

The `route_global` command also honors the hierarchical rules for plan group boundary crossings by recognizing that plan groups and the routes internal to a plan group will not cross the plan group boundaries. The routes that cross the plan groups make minimal intersections with the plan group boundaries and do not crisscross. The intersection becomes the pin location of the plan groups. You can analyze the pin assignment by checking the topology of related global routes.

Honoring the plan group boundaries and pin assignment constraints through the application of the following rules results in more accurate pin assignment and better use of valuable channel routing resources.

- Routing nets that exist only in the context of the modules associated with the plan group hierarchy, completely inside the plan group
- Not routing nets without a connection to the logic modules associated with a plan group over that plan group, unless feedthrough creation is allowed

- Ensuring that interface nets are routed across the plan group boundary at least as many times as the number of ports on those nets for the associated logic module

The router also tries to minimize the number of routes across the plan group boundary, so that they exceed the number of ports only when necessary.

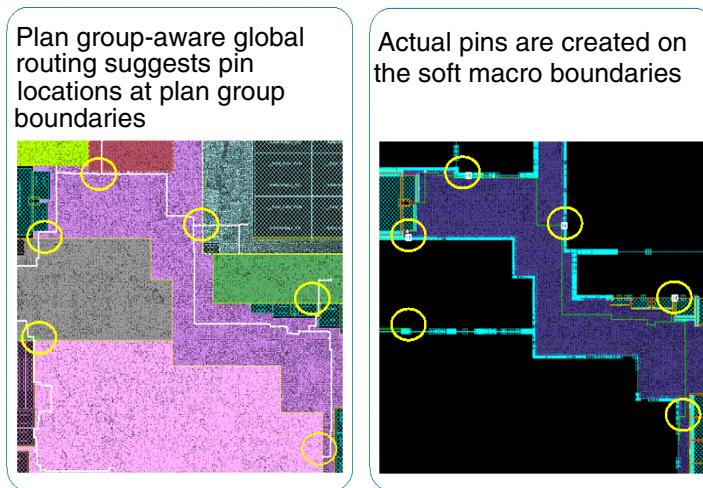
- Minimizing jogs in the channels between plan groups
- Routing nets with the clock net type first, followed by the remaining signal nets

[Figure 11-4](#) shows an example of plan group-aware global routing and pin assignment.

The picture on the left shows pins cut at the boundaries where the routes cut the plan group boundaries, as shown by the areas that are circled.

The picture on the right shows the actual pins created at the boundaries of the soft macros, as shown by the areas that are circled. These are the exact locations where the routing crosses those boundaries.

Figure 11-4 Plan Group-Aware Global Routing and Pin Assignment



Detecting Buses

If the “Keep bus bits together” option is enabled in the Add Pin Assignment Constraints dialog box, the plan group-aware router automatically detects buses based on user-defined bus groups in the Milkyway database.

Detecting Clock Nets

Before you set the pin assignment constraints, you can ensure that there are optimal locations reserved for the pin of clock nets by using the `mark_clock_tree -clock_net` command. The plan group-aware router checks the net type in the Milkyway database for any nets with the “clock” net type. Nets with the “clock” net type are routed first, followed by

the remaining signal nets. If no clocks are detected during the setting of the pin constraints, a warning message is issued saying that no clock nets were found and recommends that you run the `mark_clock_tree -clock_net` command to mark the clock nets in the Milkyway database.

Excluding Feedthroughs on Clock Nets

If you want to exclude feedthrough generation on clock nets, you can enable the “Exclude feedthroughs on clock nets” option on the Add Pin Assignment Constraints dialog box.

Alternatively, you can use the `set_fp_pin_constraints -exclude_clock_feedthroughs` command.

Performing Zroute-Based Plan Group-Aware Routing

The `set_route_zrt_common_options -plan_group_aware off | all_routing | top_level_routing_only` command supports plan group-aware Zroute global routing in the pin-cutting flow.

You should run plan group-aware Zroute global routing by using the following flow on a hierarchical placed design.

- `set_fp_pin_constraints -allow_feedthroughs on`

When this command is set to `on`, Zroute will honor the logical constraints on voltage area feedthroughs to ensure port punching on the voltage areas.

- `set_route_zrt_common_options -plan_group_aware all_routing`

When this command is set to `-plan_group_aware`, the Zroute global router becomes plan group-aware. The Zroute global router reads any plan group definitions and honors any pin placement constraints placed on those plan groups, just as the classic global router does.

Note:

If you set the `set_route_zrt_common_options -plan_group_aware` command to `off`, the Zroute global router ignores the plan groups and routes the design as “flat.” The default is `off`.

The `-all_routing` argument routes all the nets in the design and preserves the hierarchy and pin constraints for the pin-cutting flow.

- `route_zrt_global`

Run the `route_zrt_global` command to perform global routing on the design.

Performing Global Routing

After you have enabled plan group-aware routing by using the `set_fp_flow_strategy -plan_group_aware_routing true` command, use the `route_global` or `route_fp_proto` command to perform global routing to get an estimate of the pin locations and congestion of your design.

Increasing the Routing Speed

You can use the `set_fp_flow_strategy -top_level_routing_only true | false` command to increase the routing speed.

If you set the `-top_level_routing_only` option to a value of `true`, the subsequent `route_global`, `route_fp_proto` and `place_fp_pins` commands will not route any nets that are entirely contained within plan groups. Note that for this option to be effective, the `-plan_group_aware_routing` option must also be set to a value of `true`.

If you set this option to a value of `false`, the subsequent `route_global`, `route_fp_proto`, and `place_fp_pins` commands will route all signal nets.

Note:

Intraplan routing is not strictly needed for creating feedthroughs and pin locations in the subsequent `analyze_fp_routing` and `commit_fp_plan_groups` commands. However, if the routing congestion inside the plan groups is significant, ignoring intraplan group nets could lower the quality of the pin placement.

Analyzing the Pin and Feedthrough Routing

You can analyze the routing pattern generated by the plan group-aware router to determine the pin locations after global routing. (The pins are physically created during the commit hierarchy stage.) The global routing is analyzed with respect to routes crossing the boundaries of the plan groups or voltage areas. This information can be used to output a list of feedthroughs nets on plan groups or voltage areas.

Feedthroughs are created on plan groups or on hierarchical modules that belong to voltage areas.

The behavior of pin and feedthrough routing analysis is highly dependent on both the pin and voltage area constraints set previously and the last global route. (see “[Setting Pin Assignment Constraints](#)” on page 11-2 and “[Setting Voltage Area Feedthrough Constraints](#)” on page 11-8)

During routing analysis, you can view the list of nets that will have feedthroughs created for them when a design with plan groups is converted to soft macros. When you click a net in the list, all pins connected to the net are cross-highlighted in the layout (CEL) view.

To analyze the pin and feedthrough routing,

1. Choose Pin Assignment > Finalize Pin/Feedthrough Routing.

The Finalize Pin/Feedthrough Routing dialog box appears.

Alternatively, you can use the `analyze_fp_routing -finalize_pins_feedthroughs {plan groups | voltage_areas}` command.

2. Select the “Report feedthroughs only” option to output a list of all the feedthrough nets to the specified file.

If you selected the “PlanGroup” option, a list of feedthrough nets with their associated plan groups is reported. This is the default. The location, level, and direction of the pins, including the feedthrough pins, is reported.

If you selected the “Voltage areas” option, a list of feedthrough nets on voltage areas is reported.

Note:

No changes are made to the Milkyway database. The output list allows you to preview the nets whose routing implies feedthroughs on plan groups or voltage areas.

The actual layout data is not modified until you “finalize the routing,” thereby allowing you to reroute and reanalyze your design until you are satisfied with the routing results. When you are satisfied with the results, move on to step three.

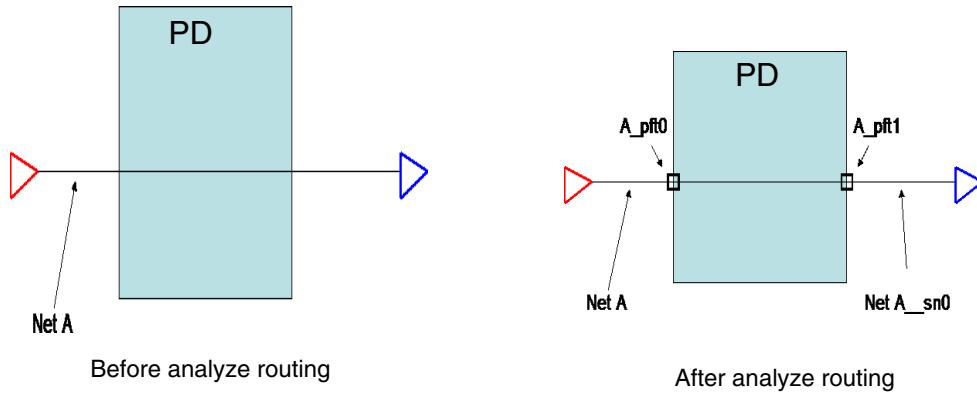
3. Finalize the routing. Choose Pin Assignment > Pin and Feedthrough Analysis.

The Feedthrough Analysis dialog box appears.

Select the Pre-Finalized Feedthrough tab and click the Finalize Pin Routing button.

- The logical netlist is modified and the feedthrough nets are updated and inserted into the hierarchy preservation, based on the existing global routing and where plan group crossings do not have corresponding connections in the hierarchy preservation. The pin locations for each boundary location are also calculated and saved in the Milkyway database as a guide to pin placement later when the hierarchy is committed.

[Figure 11-5](#) shows an example of a simple feedthrough insertion before and after analyzing the routing.

Figure 11-5 Simple Feedthrough Insertion

- If the “PlanGroup” option was selected, the pin locations for each feedthrough net crossing a plan group is also calculated and saved in the Milkyway database. This is the default behavior.
- If the “Voltage areas” option was selected, pure feedthrough nets and ports are created for nets with routes crossing between a global route and the voltage area’s boundaries. Only logical feedthrough nets and ports are created at the top level of the logical hierarchy within the voltage area.

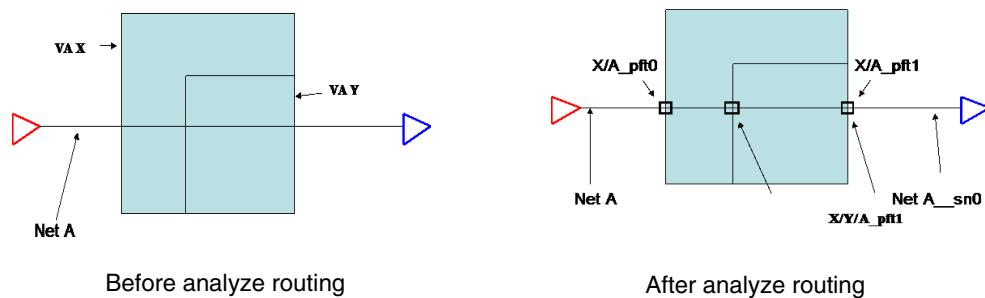
Once the feedthroughs for voltage areas are created, you can run always-on synthesis as needed to optimize the feedthrough nets.

If the routes cross a plan group which also belongs to a voltage area, a feedthrough is created on that plan group; otherwise a hierarchical module is selected within the voltage area and a feedthrough is created on the module.

Note:

There is no physical cell that corresponds to the voltage area, and therefore the physical location of the boundary crossing is not recorded.

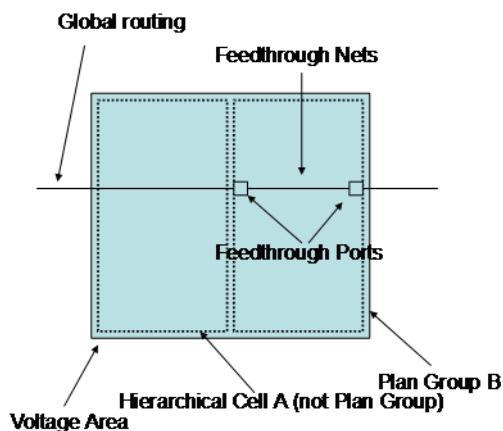
Voltage areas can be nested during feedthrough insertion. [Figure 11-6](#) shows an example of nested voltage area feedthrough insertion before and after analyzing the routing.

Figure 11-6 Nested Voltage Area Feedthrough Insertion

Voltage areas may correspond to different hierarchical cells.

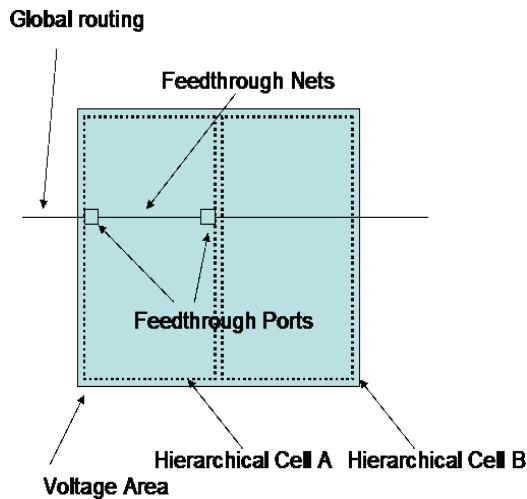
If a global route crosses a voltage area and there is a plan group within the voltage area that overlaps with the global routing, a pure feedthrough is created on that plan group and the other hierarchical cells within the voltage area remain intact. An example of this scenario is shown in [Figure 11-7](#).

Figure 11-7 Plan Group Overlaps with Global Routing; Hierarchical Cells Remain Intact



If there is no plan group overlap with global routing, a hierarchical cell that belongs to the voltage area is randomly selected, and pure feedthrough nets and ports are created in this hierarchical cell. Because the hierarchical modules inside the voltage area have no placement bounds, their cells can freely intermix. An example of this scenario is shown in [Figure 11-8 on page 11-26](#). (In this example, hierarchical cell A is selected.)

Figure 11-8 No Plan Group Overlap with Global Routing; Hierarchical Cells Intermix



During “finalize routing,” the Milkyway database is also updated by inserting feedthroughs into hierarchical modules that belong to voltage areas when the router crosses a voltage area.

Note:

After you finalize the routing, any changes made to the global routes no longer affect pin assignment.

When “finalize routing” is complete, a summary of the block names, the number of original ports on the blocks is displayed in the console log view. A text file called “VA_Feedthrough.port” is automatically written with the names of all the feedthrough ports created on the voltage areas.

Committing the Hierarchy With Cut-Pins

Pin-cutting occurs during commit hierarchy when the actual physical hierarchy, the soft macros, are created from exclusive plan groups. The soft macros use the plan group’s placement. Pin-cutting during commit hierarchy is based on plan group-aware global routing and pin constraints-aware routing results. Pin-cutting during commit hierarchy also honors the bus pin assignment constraints set by the `set_fp_pin_constraints` command.

To commit the hierarchy,

1. Choose Partition > Commit Plan Groups

The Commit Plan Groups dialog box appears.

Alternatively, you can use the `commit_fp_plan_groups` command.

2. Specify the plan groups to be committed to soft macros. The default is all.

3. Select the “Push down power and ground straps into the child cell” option if you want to push down the power and ground preroutes into the soft macros. It removes them from the top level.
4. Select the “Commit to a new top cell” option if you want to create a new top cell. Changes will occur at the top level. You must enter the name of the top-level cell.

During commit hierarchy the exclusive plan groups are converted to soft macros, and the following occurs:

- The design is converted to a two-level hierarchical design (the design netlist is partitioned to blocks and to the top level).
- CEL views are created for each soft macro.
- Nets are created in the soft macro based on the Hierarchy Preservation data.
- Cell rows (placement tiles) are cut down and pushed into the soft macro CEL views.
- (Optional) Power and ground straps and standard cell preroutes are pushed down into each soft macro CEL view.
- Placement blockages are pushed down to each soft macro.
- Pin placement is created based on the finalize routing results.

Note:

Commit hierarchy can take longer to run if power and ground straps and standard cell preroutes are pushed down because it calls repair hierarchy for each soft macro.

When pin-cutting during commit hierarchy is complete, a summary of the block names, the number of original ports on the blocks, and the feedthrough ports and nets that were created on those blocks are displayed in the console log view and the log file. This process also checks the Milkyway database for any nets with the “clock” net type and assigns the pin slots for the clock nets first. If no clocks are detected by pin-cutting during commit hierarchy, a warning message is issued saying that no clock nets were found.

Using Pin-Cutting With Multiply Instantiated Modules

The pin-cutting flow supports designs with multiply instantiated modules.

This section includes the following topics:

- [Setting Pin Assignment Constraints for Multiply Instantiated Modules](#)
- [Performing Plan Group-Aware Routing for Multiply Instantiated Modules](#)
- [Analyzing Routes and Finalizing the Routing for Multiply Instantiated Modules](#)

- [Selecting a Master Multiply Instantiated Module Plan Group](#)
- [Committing a Master Multiply Instantiated Module Plan Group](#)

Setting Pin Assignment Constraints for Multiply Instantiated Modules

Use the `set_fp_pin_constraints` command or choose Pin Assignment > Pin Constraints from the GUI to apply pin constraints for multiply instantiated modules. (See “[Setting Pin Assignment Constraints](#)” on page 11-2.) The same pin assignment constraints used in the pin-cutting flow are assigned on all instances of a multiply instantiated modules set to avoid conflicts and potential problems during plan group-aware routing, and pin-cutting during commit hierarchy. If a multiply instantiated modules instance of a multiply instantiated modules set has different pin assignment constraint settings, a warning message is issued during the `set_fp_pin_constraints` step.

Note:

No feedthroughs are allowed for multiply instantiated modules.

Performing Plan Group-Aware Routing for Multiply Instantiated Modules

The plan group-aware router performs global routing for multiply instantiated modules. (See “[Analyzing the Pin and Feedthrough Routing](#)” on page 11-22.) You can analyze the results for each plan group based on wire length, timing, and congestion reports. Based on this analysis, you should be able to determine which of the multiply instantiated modules plan groups to use as the master instance in the multiply instantiated modules set.

The plan group-aware router automatically turns off feedthroughs on multiply instantiated modules blocks.

Note:

If any multiply instantiated or unique modules are instantiated as CEL views in the design and pin assignment has not been run on them, the plan group-aware router issues an error message.

Analyzing Routes and Finalizing the Routing for Multiply Instantiated Modules

Run the `analyze_fp_routing` command or choose Choose Pin Assignment > Finalize Pin/Feedthrough Routing from the GUI to analyze the routing pattern generated by the plan group-aware router. (See “[Analyzing the Pin and Feedthrough Routing](#)” on page 11-22.)

When you are satisfied with the routing results, choose Pin Assignment > Pin and Feedthrough Analysis. Select the Pre-Finalized Feedthrough tab and click the Finalize Pin Routing button to finalize the routing.

The `analyze_fp_routing` command determines the pin locations on the multiply instantiated modules during commit hierarchy based on the plan group-aware routing results for the selected master instance.

Note:

If any multiply instantiated or unique modules are instantiated as CEL views in the design and pin assignment is run using the `place_fp_pins` command, finalize routing considers their pin placement as fixed and finalizes the routing for the remaining plan groups.

Selecting a Master Multiply Instantiated Module Plan Group

Before committing a master multiply instantiated module plan group, you must first choose a master plan group by using the `select_mim_master_instance` command to designate a plan group as the “master” (template) plan group among a set of multiply instantiated module plan groups that will use the same cell instance master for all multiply instances.

The syntax is:

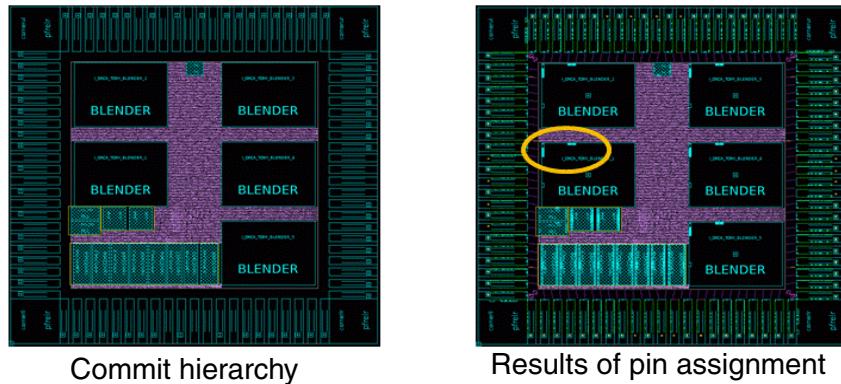
```
select_mim_master_instance plan_group
```

Use the `plan_group` argument to specify which plan group among a set of multiply instantiated module plan groups the `commit_fp_plan_groups` command should treat as the master plan group.

A multiply instantiated module group is a set of plan groups that retain a common database property. This database property associates the plan groups with each other and identifies the name of the hierarchical cell instance reference. When a multiply instantiated module group exists with this common property, the `commit_fp_plan_groups` command identifies one plan group among this multiply instantiated module group as the master plan group and then only creates a CEL view once for the plan group designated as the master of its multiply instantiated module group.

[Figure 11-9 on page 11-30](#) shows pin assignment results after committing the hierarchy.

Figure 11-9 Pin Assignment After Commit Supports Multiply Instantiated Modules



Note:

If you do not select an multiply instantiated module master plan group before committing the hierarchy, the `commit_fp_plan_groups` command issues a warning and then randomly chooses the first multiply instantiated module plan group in the multiply instantiated module list as a master plan group.

The following example specifies a particular plan group as the master of its multiply instantiated module group.

```
select_min_master_instance [get_cells {planGroupA}]
```

The set of multiply instantiated module plan groups was previously grouped by the `uniquify_fp_mw_cell -store_mim_property` command. (See “[Determining Which Plan Groups and Soft Macros are Multiply Instantiated Modules](#)” on page 5-44.) This command creates multiply instantiated module plan groups that are identified as sharing a common hierarchical reference cell.

The master instance information is saved as a plan group property in the Milkyway database.

Committing a Master Multiply Instantiated Module Plan Group

Use the `commit_fp_plan_groups` command to convert a master multiply instantiated module plan group to a soft macro. Other multiply instantiated module plan group instances from the same multiply instantiated module set are also converted to instances of the master soft macro at the same time.

Pin-cutting occurs during commit hierarchy for multiply instantiated modules blocks based on the pin locations determined by the `analyze_fp_routing` command for the chosen master instance and the plan group-aware routing results. Pin-cutting supports multiply instantiated modules blocks mirrored along both the x- and y-axes.

Performing Incremental Pin-Cutting

You can incrementally assign pins in the pin-cutting flow.

To perform incremental pin-cutting,

1. Perform pin-cutting on a group of selected nets by using the `set_fp_pin_constraints -nets` command.
2. Perform pin-cutting again on a second group of nets, or on the rest of the nets, by using the `set_fp_pin_constraints -incremental` on `-nets` command.

IC Compiler appends the results of the pin locations from the second (or final) group of nets to the already calculated results of the first group of nets.

By default, performing pin-cutting (`analyze_fp_routing` command) overrides all previously recorded pin locations.

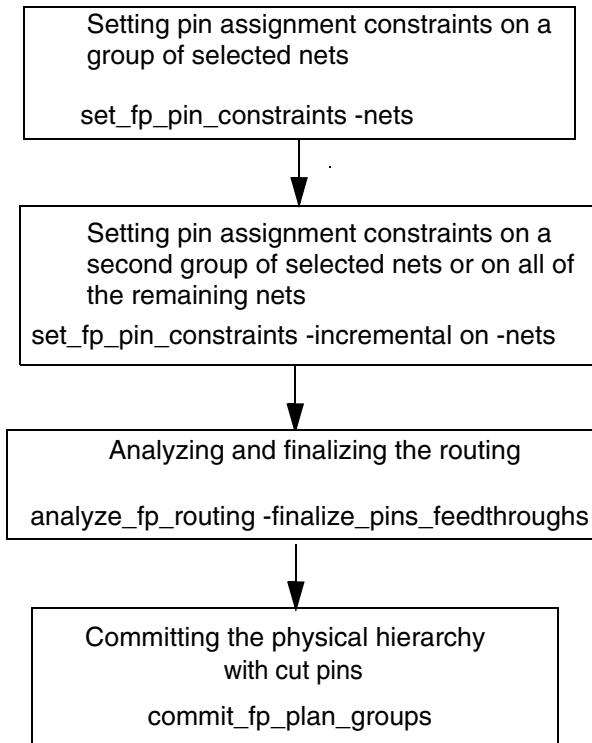
Note:

When performing incremental pin-cutting, keep in mind that although it provides you with a lot of flexibility, and the ability to place pins incrementally, a cost is also incurred because it becomes necessary to run global routing (`route_global` command) and analyze routing (`analyze_fp_routing` command) multiple times.

This section includes the following topics:

- [Setting Pin Assignment Constraints on a Group of Selected Nets](#)
- [Setting Pin Assignment Constraints on a Second Group of Selected Nets or on All Remaining Nets](#)
- [Analyzing and Finalizing the Routing on a Separate Group of Nets](#)
- [Committing the Hierarchy With Cut Pins](#)

[Figure 11-10 on page 11-32](#) shows the incremental pin-cutting flow.

Figure 11-10 Incremental Pin-Cutting Flow

Setting Pin Assignment Constraints on a Group of Selected Nets

Set the pin assignment constraints for the first group of selected nets on which pin-cutting will be run. These nets are processed and pin locations are recorded in a pin-cutting attach file.

```
icc_shell> set_fp_pin_constraints -nets
```

Setting Pin Assignment Constraints on a Second Group of Selected Nets or on All Remaining Nets

Change the pin assignment constraints to indicate pin-cutting will be run on a second, separate group of nets, or on all of the remaining nets. Turn on the `-incremental` option. When this option is on, existing soft macro pins are retained.

```
icc_shell> set_fp_pin_constraints -incremental on -nets
```

Analyzing and Finalizing the Routing on a Separate Group of Nets

Analyze the existing global routing to determine the pin locations and feedthroughs needed on each plan group by using the `analyze_fp_routing` command. The global routing is analyzed with respect to routes crossing the plan groups. The nets should now be considered fixed.

When you are satisfied with the results, finalize the routing.

```
analyze_fp_routing -finalize_pins_feedthroughs {plan_groups |  
voltage_areas}
```

The second group of selected nets is finalized, and the pin locations are added to the first pin-cutting attach file. The attach file now contains the pin locations for both groups of nets.

IC Compiler appends the results of the pin locations from the second (or final) group of nets to the already calculated results of the first group of nets.

Committing the Hierarchy With Cut Pins

Pin-cutting occurs during commit hierarchy (`commit_fp_plan_groups` command) when the actual physical pins are created, and is based on plan group aware global routing. The `commit_fp_plan_groups` command converts the specified plan groups into soft macros. For each plan group that is processed, a soft macro is created to replace it.

Creating Rectangular or Rectilinear Pin Guides

You can create rectangular or rectilinear pin guide shapes (pin keep-in regions) on soft macro boundaries, on plan groups, or on the current top-level cell to control the placement of pins or ports on soft macros or plan groups.

This section includes the following topics:

- [Accessing Information About the Pin Guides](#)
- [Creating Pin Guides for Feedthroughs](#)

Because the plan groups do not yet have ports, these constraints are applied to nets rather than to individual pins. The pin assignment engine (`place_fp_pins` command) will then determine the most appropriate location for those pins in the pin guide (keep-in region) based on routability and wire length.

Note:

Pin guides must intersect the module boundary in one contiguous area. Multiple intersections imply that you want a feedthrough on the net. If there are two or more external connections for the net, it is interpreted as a “real” feedthrough, that is, one of the ports is the original port on the block, and the other is the new port.

The routing for the nets that have ports associated with the pin guides is constrained to cross the corresponding soft macro or plan group boundaries through the pin guide regions. As a result, pin assignment honors the pin guide regions and places pins within the corresponding soft macro edge segment. If the pin guide overlaps more than one edge of the soft macro, the router can choose to go through any edge segment that overlaps that pin guide.

The pin guide is a polygon associated with a group of ports or pins and the soft macro or plan group it overlaps with. Only one pin guide can be associated with any given pin on a soft macro or plan group. If the pin guide does not overlap any soft macro or plan group, or if it does not have any ports associated with it, it is disregarded by the router and the pin assignment engine.

To create a pin guide,

1. Choose Pin Assignment > Create Pin Guide.

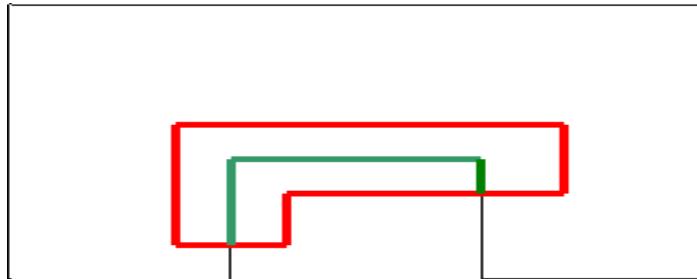
The Create Pin Guide dialog box appears.

Alternatively, you can use the `create_pin_guide` command.

2. In the Name text box, enter the name of the pin guide you want to create. If you do not use this option, a default name is given to the newly created pin guide.
3. Select the object (soft macro, plan group, black box, or top cell) on which to create a pin guide. The default is soft macro. Enter the name of the soft macro, plan group, black box or top cell in the text box.
4. Under the “soft macro pins” heading, select the type of guide (Pin, Nets, or Ports) and enter a comma, or space-separated list of pins, nets, or ports.
5. Specify the rectangular or rectilinear boundary points of the rectangular or rectilinear pin guide. You can enter the coordinates of the corner points or draw the area graphically in the layout window.

For a rectilinear plan group, you can draw a rectilinear shape to define a pin guide, as shown in [Figure 11-11 on page 11-35](#). The pin may reside anywhere within the green area.

Figure 11-11 Rectilinear Shape Defining a Pin Guide



Accessing Information About the Pin Guides

You can access information about specific pin guides by using the `report_pin_guides` command. Select the `-pins` option to get detailed information about pin guides associated with specified pins or select the `-nets` option to get detailed information about pin guides associated with specified nets.

Creating Pin Guides for Feedthroughs

You can specify the locations of feedthroughs on a block (soft macro or plan group) for a given net by creating pin guides for the feedthroughs.

Note:

The feedthrough guide only controls where the pins are placed on the block boundaries. It does not control the routing inside the block.

If a pin guide shape passes through a soft macro or plan group, and if the net does not already touch the soft macro or plan group, a “pure” feedthrough is created. A “pure” feedthrough is unconnected internally; it has no connection in the block it passes through, except for the ports of the block. Conversely, if the feedthrough net has one or more cell instances within the block (soft macro or plan group) that the net is connected to, then one side of the feedthrough is counted as the original port, and the other side is a “regular” (connected) feedthrough. To ensure that the connectivity of the feedthrough is clear, the pin guide shape should touch the port, plan group, or soft macro that is connected to the net.

Feedthrough generation takes place only under the following two conditions:

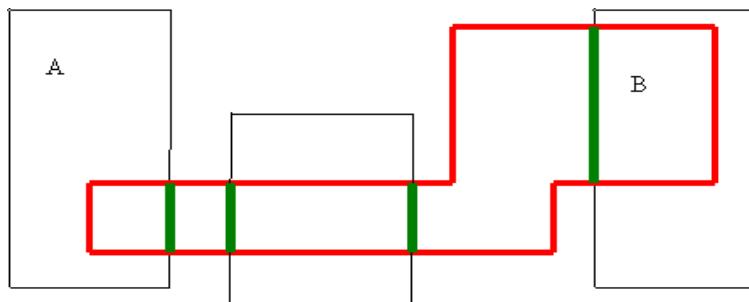
- The `allow_feedthroughs` option must be selected (select the “Feedthrough Settings” tab on the Pin Assignment Constraints dialog box).

You can also specify the `set_fp_pin_constraints -allow_feedthroughs` on command.

- The block must be identified in the list of “parents” (`create_pin_guide -parents object`) when the pin guide is created.

As shown in [Figure 11-12](#), port A on the left plan group connects to port B on the right plan group. The center plan group does not contain any ports in the net. The pin guide shape (shown in red) will constrain the pins on the left and right plan groups to the green areas and will cause feedthroughs to be created on the center plan group in the green areas. It is not necessary to include the actual location of the instance ports in the plan groups in the pin guide shape.

Figure 11-12 Pin Guide for Feedthrough



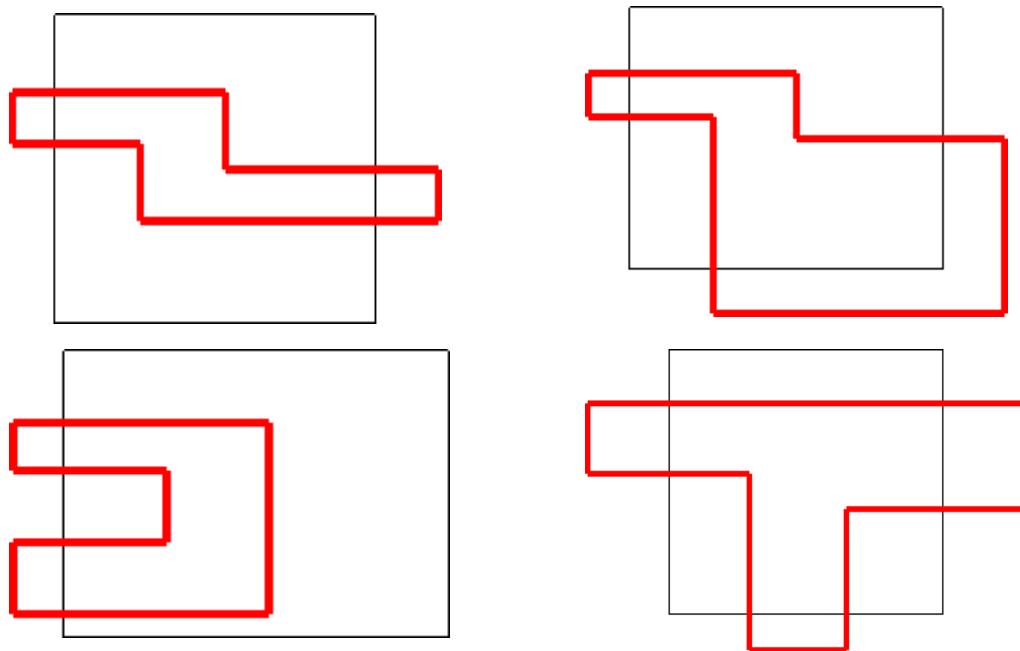
Supported Feedthrough Guides

IC Compiler supports the following rectilinear feedthrough guide shapes, as shown in [Figure 11-13 on page 11-37](#).

A feedthrough guide on a block must have at least two connections on the net outside the block. If it does not, the feedthrough guide is ignored and a warning message is issued.

A feedthrough guide must intersect the module boundary in two disjoint areas.

Figure 11-13 Rectilinear Feedthrough Guide Shapes



Performing Pin and Feedthrough Analysis

You can inspect the physical paths of nets between their endpoints and through the hierarchy by tracing and highlighting the connectivity of specified top-level soft macro nets and pins, and the complete feedthrough path of all feedthrough nets and pins generated for plan groups and soft macros in your design.

You can perform feedthrough analysis on voltage areas and buffered feedthrough nets that are split by inserting buffers during timing optimization.

You can explore detour nets, including both pin detour severity and route detour severity.

You can display all the nets that belong to the traced path and cross-highlight them in the layout window. You can also view multiple nets at the same time, using different colors to do topology analysis.

To perform soft macro net-based or pin-based feedthrough analysis,

1. Choose Pin Assignment > Pin and Feedthrough Analysis.

The Feedthrough Analysis dialog box appears.

You can use the Feedthrough Analysis dialog box to

- Trace all the nets for a path or all the feedthrough nets in the design
 - Highlight multiple nets and use different highlight colors to view net locations in the layout view
 - Select nets you want to gather information about by using the Query tool
 - Explore the physical path of a net by coloring individual net segments or pins in the layout view
2. Select the Finalized Feedthroughs tab and click the Analyze Nets button.
- The Analyze Nets dialog box appears.
3. Select an analysis mode option.
- To analyze feedthrough nets for plan groups and soft macros, including black boxes, select the Blocks option.
- To analyze feedthrough nets for voltage areas, select the “Voltage areas” option.
- To include feedthrough nets that are split into multiple sections due to the introduction of buffer cells during timing optimization, select the “Buffered feedthrough” option.
4. Click the Analyze Nets button
5. Identify the original plan group nets or soft macro nets or pins you need to trace by doing one of the following:
- To trace all the nets in the design, select the “All nets” option.
 - To trace all the feedthrough nets in your design, select the “All Feedthrough nets” option. This is the default.

To trace feedthrough nets only for specific blocks, select the “Specified blocks” option and either select or browse for the block or type their names in the “Specified blocks” text box. If you enter more than one name, use a space or a comma to separate the names.
 - To trace all the non-feedthrough nets in your design, select the “All non-feedthrough nets” option.

To trace non-feedthrough nets only for specific blocks, select the “Specified blocks” option and either select or browse for the block or type their names in the “Specified blocks” text box. If you enter more than one name, use a space or a comma to separate the names.
 - To trace all the bus nets in your design, select the “All bus nets” option.

To trace bus nets only for specific blocks, select the “Specified blocks” option and either select or browse for the block or type their names in the “Specified blocks” text box. If you enter more than one name, use a space or a comma to separate the names.

- To trace one or more nets, select the “Specified nets” option and either select or browse for the nets or type the net names in the “Specified nets” text box.
- To trace the net connections for one or more pins, select the “Specified pins” option, and either select or browse for the pins or type the pin names in the “Specified pins” text box.
- To trace the nets through voltage areas instead of plan groups, select the “Specified voltage areas” option and type the voltage area names in the “Specified voltage areas” box.

6. Click OK in the Analyze Nets dialog box.

The tool filters the net or pin connections and displays the names of the original top-level nets in the Names nets list. These are the nets that existed in the design prior to design planning and pin assignment before they were split into several internal and external soft macro nets.

7. (Optional) If you want to display status information for the nets, select the “Show status columns” option. The list displays the information in the following columns:

Pin detour severity
Original feedthroughs
Pure feedthroughs
Regular feedthroughs

8. Select the names of one or more nets in the nets list that you want to highlight or select.

You can click on an individual name in the list or use Shift-click or Control-click to select combinations of names.

By default, the current highlight color red is applied to the nets in the layout window.

- To highlight the nets when the “Auto color” option is not selected, click the Color button.

Highlighted nets that are not selected appear in the current highlight color in the layout view. The highlight color also appears beside the net name in the Net names list.
- To magnify the design and fit the nets in the view when the “Auto zoom fit” option is not selected, click the Zoom Fit button.
- To select the nets, click the Select button.

The selected nets appear in the selection color in the layout view. The default is white.

9. Select options to determine the appearance of the active layout view.

- Auto color

Automatically highlights the nets in the active layout view that correspond to the original net names in the display field. The default is on.

When you click on a feedthrough net, all net segments from the driver to the receiver, all pins connected to nets, and any in-place optimization buffers in between, are cross-highlighted in the active layout view.

- Auto zoom fit

Automatically zooms in on the displayed nets you highlight in the layout view. The default is off.

- Exclude Child Nets

When you select this option, child nets are excluded; otherwise nets are drawn inside soft macros.

If the net is a feedthrough net, it is drawn from the driver at the top level, through one or more soft macros, to the receiver. If the net ends at the boundary of a soft macro, the net inside or outside the soft macro is also drawn.

10. Click the Explore Net button.

The Explore Displayed Nets dialog box appears.

When the Feedthrough Analysis dialog box is open and the names of the original top-level nets in the Nets list, located below the Analyze Nets button are displayed, you can use the Explore Nets dialog box to explore the physical paths for individual nets in the active layout view.

The dialog box displays information about each net segment, including the name of the original net, the cell name, and the pin direction of the feedthrough path.

You can click the plus sign or minus sign expansion button next to a net segment name to display or hide the names of its pin connection.

11. Click Close to close the Feedthrough Analysis dialog box.

Removing Feedthrough Ports and Nets From Blocks and Voltage Areas

You can remove the feedthrough ports and nets that exist in your design, including those created by other tools, from all soft macros and plan groups in the design or on a per net basis. You can also remove feedthrough ports from voltage areas and feedthroughs with buffers inserted inside hierarchical blocks (soft macros or plan groups) or voltage areas.

For pure feedthroughs, all ports and the child level net are removed. For regular feedthroughs, one port will remain as well as the child level net.

When a feedthrough port is removed, all the pins of that port are deleted and the nets (for which feedthroughs are removed at the parent level of the logic hierarchy) are reconnected.

To remove feedthrough ports and nets,

1. Choose Pin Assignment > Remove Feedthroughs.

The Remove Feedthrough Ports and Nets dialog box appears.

Alternatively, you can use the `remove_fp_feedthroughs` command.

2. Remove feedthrough ports and nets – Choose whether to remove route feedthrough ports and nets from all plan groups and soft macros in your design (the default), or from a specified plan group or soft macro.

To remove feedthrough ports and nets from a specified plan group or soft macro, click in the text box and enter the name of the plan group or soft macro from which to remove the feedthrough ports.

3. Remove feedthrough ports from nets – Choose whether to remove feedthrough ports from all top-level nets in the design (the default), or from specified nets. To remove feedthrough ports from specified nets, click in the text box and enter the name of the net from which to remove the feedthrough ports.
4. Remove feedthrough ports from voltage areas – Choose whether to remove feedthrough nets and ports from all voltage areas or from specified voltage areas in your design. The default is all.
5. Remove original and buffered feedthroughs – When the original option is selected, the tool is instructed to remove all feedthrough ports and nets that are determined to be original as well as the added ports and nets. Feedthrough ports with an original property are defined as all feedthrough ports on blocks (plan groups and soft macros) that were generated originally from a Verilog definition. The original feedthrough ports and their connections to the top-level nets are removed.

When the buffered option is selected, the tool is instructed to remove the buffered feedthrough ports that are determined to be added as well as the unbuffered feedthroughs. Feedthrough ports with an added property were automatically generated by pin assignment (`fp_place_pins` command) or by pin-cutting (`analyze_fp_routing` command), or were created manually. The inserted buffers are removed along with the buffered feedthrough ports.

Note:

You can select both options together to remove both original and buffered feedthrough ports and nets.

If this option is disabled, the placement of original feedthrough nets and ports on blocks and voltage areas and their top-level connections are retained and the added buffered feedthrough nets and ports from all nets on all blocks and voltage areas are removed.

6. Click OK or Apply to remove feedthrough nets and ports from your design.

The feedthrough ports are removed as they are processed. The feedthrough nets are removed after the last feedthrough port of a net is removed. The hierarchy preservation is updated on-the-fly both inside the blocks and at the top level.

Refining the Pin Assignment

You can analyze and evaluate the quality of the pin assignment results by checking the placement of soft macros pins in the design and the pin alignment. You can then refine the pin assignment based on the results.

This section includes the following topics:

- [Checking the Pin Assignment](#)
- [Checking the Pin Alignment](#)

Checking the Pin Assignment

You can check or verify the placement of soft macro pins in a design. Once pins have been assigned, added, or moved, you can quickly verify if there are any spacing errors or missing pins. The relevant constraint values are read from the Pin Assignment Constraints dialog box (`set_fp_pin_constraints` command) to perform the pin placement checks.

To check the placement of soft macro pins,

1. Choose Pin Assignment > Check Pin Assignment.

The Check Pin Assignment dialog box appears.

Alternatively, you can use the `check_fp_pin_assignment` command.

2. Choose whether perform pin checking within all soft macro pins or within specified soft macro pins. The default is “All”. Checks are applied to pins on all soft macros in the design.

If you want to perform checks on top-level pins, select the “Block level” option. By default, checks are performed on soft macro pins.

3. Choose whether to perform pin placement checks only on soft macro pins and relevant hard macro pins that are connected to the nets you specify. If you select the “Specified” option, enter a collection of net names in the text box. By default, checks are performed on all relevant soft macro pins and hard macro pins.

4. Set the remaining options as needed. You can check for:

- Spacing between signal pins – Checks if the spacing between any two signal pins exceeds the number of the wire tracks you defined in the “Pin spacing” option (Pin Assignment Constraints dialog box) or the `-pin_spacing` option (`set_fp_pin_constraints` command).

IC Compiler reports an error for pins that are closer than the given number of wire tracks to a neighboring pin. The default is 1.

- Spacing between preroute and pins – Checks if the spacing between pin and prerouted net objects exceeds the number of wire tracks you defined in the “Pin preroute spacing” option (Pin Assignment Constraints dialog box) or the `-pin_preroute_spacing` option (`set_fp_pin_constraints` command).

IC Compiler reports an error for pins that are closer than the given number of wire tracks to a nearby preroute. The default is 3.

If the design has wide (“fat”) wires, the tool checks the distance between the wire to the pin on the same layer. It reports an error if the distance is smaller than the minimum distance found in the fat wire spacing table for the given layer of the wire and the width of the wire.

- Pin overlaps on same layers – Checks for shorts. A short occurs if two pins on the same layer touch or overlap each other.
- Pin overlaps across layers – Reports the overlapping pins on different metal layers.
- If pins are centered on wire track – Checks if all pins are centered on a wire track.
- If signal pins are placed on legal layers – Checks if all signal pins are placed within a predefined range on legal metal layers as specified in the “Allowed layers from” option (Pin Assignment Constraints dialog box) or the `-allow_layer` option (`set_fp_pin_constraints` command).

You must specify a minimum and maximum range for the available metal layers. If a range is not specified, no checking is performed.

- If power/ground pins are placed on legal layers – Checks if power and ground pins are placed on legal layers.
- Any missing signal pins – Checks for any missing signal pins.
- Any missing power/ground pins – Checks for any missing power or ground pins.
- Single pins on abutted edges – Checks for any unroutable pins on abutted edges. A macro pin is flagged as a single pin (unroutable) if any of one of the following scenarios occurs.

The macro pin is placed on a part of a macro edge that abuts another macro edge, and the macro pin does not abut the other pin on the same metal layer to which it is connected.

The macro pin is placed on a part of a macro edge that coincides with the top-level cell boundary and the macro pin is connected to anything other than a terminal (top-level pin), unless the pin is connected solely to other pins on the same macro, and those pins are stacked on top of the macro pin.

The macro pin is placed on a part of a macro edge that coincides with the top-level cell boundary and the macro pin is connected solely to a terminal, but the terminal is not stacked on top of the macro pin on the same metal layer.

5. Select OK or Apply.

If errors occur during the pin checking, first check your constraints to make sure you have captured them correctly and then check the log files to make sure you have applied them correctly. If the pin checks are good, but the pin placement still has errors, contact SolvNet online customer support or the Synopsys Technical Support Center for assistance.

Checking the Pin Alignment

You can check the pin alignment quality of results (QoR). You can view a list of nets with pins that are not optimally aligned by pin assignment. The list displays the total number of two connection nets among the specified nets that have at least one of their two connections inside a soft macro cell. It also reports the number and percentage of misaligned nets among the two connection nets.

You can also generate a detour report for pin feedthroughs.

To check the pin alignment,

1. Choose Pin Assignment > Check Pin Alignment.

The Check Pin Alignment dialog box appears.

Alternatively, you can use the `check_fp_pin_alignment` command.

2. Choose whether to check the pin alignment on all pins (the default) or on specified pins.

3. You can generate a detour report for pin feedthroughs. Choose the “Check pin feedthroughs detour” option.

A pin detour report displays the net where the bounding box of the nonsoft macro pins on the net is smaller than the bounding box of all the pins on the net.

You must specify a tolerance value for the detour report. The valid range is between 0 percent and 100 percent. The default value is 10 percent.

If you do not specify the detour option, normal pin alignment checking is performed.

4. Select OK or Apply.

12

Performing Timing Budgeting

During the design planning stage, timing budgeting is an important step in achieving timing closure in a physically hierarchical design. The timing budgeting determines the corresponding timing boundary constraints for each top-level soft macro or plan group (block) in a design. If the timing boundary constraints for each block are met when they are implemented, the top-level timing constraints are satisfied.

Timing budgeting distributes positive and negative slack between blocks and then generates timing constraints in the Synopsys Design Constraints (SDC) format for block-level implementation.

Timing budgeting distributes the timing constraints from the top level to the block level, creating a new constraint set for each of the top-level blocks in the design. Budgeting also allocates slack between blocks for timing paths crossing the block boundaries. Timing budgeting propagates the timing constraints downwards one hierarchical level at a time. Timing budgeting also propagates timing exceptions to block-level constraint sets.

You can use the timing budgeting feature in virtual flat mode to generate constraints for plan groups.

During timing budgeting, you can also create quick timing models for soft macros and plan groups and load them into your design.

This chapter includes the following sections:

- [Timing Budgeting Prerequisites](#)
- [Performing Pre-Budget Timing Analysis](#)
- [Running the Timing Budgeter](#)
- [Generating a Quick Timing Model From a CEL View](#)
- [Performing Timing Budgeting On Plan Groups](#)
- [Performing Hierarchical Signal Integrity Budgeting](#)
- [Performing Post-Budget Timing Analysis](#)
- [Performing Clock Latency Budgeting](#)

Timing Budgeting Prerequisites

Before a cell can be budgeted, it has to meet the following requirements:

- It must be floorplanned and all cells must be placed and legalized. The closer the floorplan is to the final layout, the better the budgeting results will be.

Note:

A globally routed design, while not required, results in budgets that are more accurate.

- The top-level timing constraints must be loaded.
- The timing information available must be sufficient to successfully execute the timing analyzer on the cell to be budgeted. This requires complete timing views for all top-level soft and hard macros.

You can run budgeting on a design that does not have timing models for all soft macros. If a soft macro does not have a timing model, the paths that interface with that macro are automatically marked as false paths.

- The timing information available must be sufficient to successfully execute the timing analyzer on the cell to be budgeted. This requires complete timing views for all top-level soft and hard macros.

You can run budgeting on a design that does not have timing models for all soft macros. If a soft macro does not have a timing model, the paths that interface with that macro are automatically marked as false paths.

- Run a timing report on your top-level design before performing timing budgeting. If design errors are reported during the timing report, budgeting cannot be run successfully.
- Timing paths should have reasonably small slack values.

Budgeting is effective only when your design is close to meeting timing. It would be very difficult to meet timing if, for example, the worst negative slack was 10 nanoseconds and the clock period was also 10 nanoseconds. Timing is considered reasonable, however, if the worst negative slack is 10 percent or less but no more than 20 percent of the clock period. If it is more than 20 percent of the clock period, you should carefully review the results of the path timing report to determine the cause of the timing violations.

The most common cause of incomplete or missing constraints in the output timing boundary files is that no timing path passes through the port for which constraints are either incomplete or missing. This may be intended if false path specifications are loaded. However, it may be caused unintentionally, for example, by incomplete or incorrect timing views or by dangling nets.

Performing Pre-Budget Timing Analysis

You can perform pre-budget timing analysis on your design. The timing analysis is applied to blocks (plan group or soft macro) or the top-level design. Pre-budgeting timing analysis allows you to handle early checking of the timing rules and constraints set in the Synopsys Design Constraints (SDC) file before your design is submitted to the budgeting process.

By providing feedback on design and timing information, the timing checking analysis tool can help you determine the feasibility of your design and its timing constraints.

You can generate a report file that contains additional design information for validating the budgets assigned to each block in your design. You can use this report to analyze why certain budgets are assigned to each block port and to debug the budgets generated by the IC Compiler design planning timing budgeter.

The design information in the report file includes, but is not limited to, clocks in the design, timing constraints in the design, timing exceptions in the design, delay distribution on the worst timing path through the block ports, pins tied to high/low, high negative slack on the timing path, and so on.

To generate a pre-budgeting timing analysis report file,

1. Choose Timing > Floorplan Timing Environment Check.

The Timing Environment Analysis Options dialog box appears.

Alternatively, you can use the `check_fp_timing_environment` command.

2. Select the Budgeting Analysis tab. This is the default.
3. Enter the hierarchical cell name or plan group name and the pin names on which to apply the budgeting analysis.

4. Select the options to use for performing budgeting analysis. The options you specify determine the type of information that is written to the output report file.

You can show reports on:

- Block pin statistics – For each block, prints the number of hierarchical pins that can and cannot be budgeted and the total number of pins on the block. A hierarchical pin is budgeted if timing constraints or exceptions are propagated to it. The default is on.
- Unbudgetable pins – Reports why pins cannot be budgeted. The default is on. A hierarchical pin might not be budgeted for the following reasons:

The pin is not connected to a net.

The pin is not connected to a top-level net, but is connected to an internal net.

The pin is not connected to an internal net, but is connected to a top-level net.

Only fixed delay exists on one side of the pin. (There is nothing to budget but there will still be an input or output delay constraint on these pins.)

Timing paths through pins are not constrained.

The pin is constrained by a user-defined budget.

The pin is connected to a power or ground net, or it has a `set_case_analysis` constraint on it.

The pin is either an input or output clock port.

- Unconstrained pins – For each block to be budgeted, lists the pin names that are not constrained. A pin is unconstrained if no timing path goes through it. The default is on.
- Exception pins – For each block to be budgeted, lists the pin names that have timing exceptions propagated to it. The exceptions are `set_false_path`, `set_multicycle_path`, `set_min_delay`, or `set_max_delay`. The default is on.
- Static logic pins – For each block to be budgeted, lists the pins that are set to a static logic state as a result of a case analysis statement. It reports if a block pin is tied high (1) or low (0) by a power or ground net in the logical netlist. It also reports if a block pin is set to a specific state by `set_case_analysis`, `set_logic_one`, or `set_logic_zero` constraints. The default is on.
- Number of pin connections – Reports the number of pin connections from the startpoint and the endpoint of the timing paths going through one block to other blocks, through top-level standard cells, through pad cells, through top-level ports, and through ports on the plan group.

Enter the number of pin connections to display for each type of net connection. The default is 1.

- Delay violating cells – Lists the top-level cells of interface paths that have cell delays greater than the specified percentage of the captured clock period. It traverses all the critical paths per clock domain through the block pin.

The *percent* argument is required and must be greater than or equal to zero.

- Report to file – Prints the report to an output file in a single-line format. Enter the name of the output file. The default is off.
- Output in table format – Formats the report in a table. If the report is in a table, it is easier to read, but it may be difficult to parse using automated scripts. The default is off.

5. Select the Timer and Bottleneck Analysis tab to generate timing and bottleneck reports.

You can show timing and bottleneck reports on:

- Zero wire delay – Performs zero wire delay analysis and reports the negative-slack paths that have slack less than the negative percentage of the captured clock period. This will call timing analysis in a special mode, in which the wire delays are set to zero. The default is off.

Specify a slack limit. The slack limit is the threshold percentage of the end clock period for each negative-slack path. A timing path is printed in the output report file if its slack is less than the negative percentage of the path's captured clock period.

- Virtual IPO – Reports timing analysis results based on virtual in-place optimization. The default timing report from this option uses a slack threshold of zero. The default is off.

Specify a slack limit for the timing paths in the virtual in-place optimization timing report. Only paths that have slack less than the slack percentage of the capture clock period are reported.

- Bottleneck cells – Reports bottleneck cells. It lists the cells in the design that contribute to multiple timing violations. It lists the leaf cell, its reference, and the number of paths through the leaf cell that have timing violations. Based on this report, you can check the fanin and fanout logic to determine the possible cause of the timing bottleneck. The default is off.
- Run Virtual IPO – Disable this option if you do not want virtual in-place optimization to run before reporting bottleneck cells. The default is to run virtual in-place optimization.
- Slack limit – You can specify the slack limit for reporting bottleneck cells only. Only timing paths having slack less than the slack limit will be explored to locate bottleneck cells. The default is 0.
- Maximum number of cells – You can specify the maximum number of bottleneck cells to report. This allows you to limit the number of bottleneck cells reported for a design. The default is 20.

- Maximum number of paths – You can optionally specify a maximum number of paths to report.

6. Click OK.

The following information is printed in the report file for the whole design:

- Negative-slack paths based on zero wire delay
- Bottleneck cells
- Timing report based on virtual in-place optimization

The remainder of the report is divided by soft macro or plan group. All the timing environment information related to one soft macro or plan group is printed together.

Running the Timing Budgeter

You can run the timing budgeter to perform proportional timing budgeting (the default) or advanced timing budgeting based on virtual in-place optimization for plan groups or soft macros in the design.

Proportional budgeting allocates positive and negative slack based on the actual physical circuits in each path. It also allocates slack on the path based on the delay on the path portions within individual blocks. Timing budgets are distributed proportionally to the worst or best case delay that is actually present for each path (per clock domain) inside the top-level soft macros. Portions of interface paths which are hierarchically in the top level are considered fixed delays just as hard macros are considered as fixed delays. This means that all slack for interface paths is budgeted solely to blocks.

After slack is allocated between blocks, the design is processed to generate a timing constraints file in the Synopsys Design Constraints (SDC) format for each block that has a timing model.

To run the timing budgeter,

1. Choose Timing > Allocate Budgets.

The Allocate Budgets dialog box appears.

Alternatively, you can use the `allocate_fp_budgets` command.

2. Set the options, depending on your requirements.

- Hierarchical cells – Select this option to specify a list of hierarchical cell names for budgeting. By default, all plan groups and soft macros in the current design are budgeted. The default is off.

- Fixed delay plan groups – Select this option if you want a list of cells to be treated as hard macros for budgeting purposes. This means that the budgets (input and output delays) calculated for cells designated as fixed delay cells will not have any slack allocated to them. This effectively allocates any time budget, which would normally be distributed to the fixed delay cell, to other soft macros which share in any interface timing paths associated with the fixed delay cell. The default is off.

Enter the names of the cells you want to be treated as fixed delay cells in the text box.

- Fixed delay objects – Select this option to specify that certain objects that are implemented have fixed delays that will not change in subsequent optimizations. The list of objects includes plan groups, soft macro cells, hard macro cells, black boxes, hierarchical cells for plan groups, pins of soft macros with interface logic models, and pins of hierarchical cells for plan groups.

The delay allocated to the timing paths passing through these pins during budgeting will be equal to the current delay.

- Black box cells – Select this option if you want a list of black box cells to be budgeted with black box timing models. The default is off.

Enter the names of the black box cells in the text box. Black box cells are not budgeted unless they are included in this text box. Black box cells with timing models which are not included in the text box are treated by budgeting the same way as hard macros are treated.

- Output file name format specification – Enter the name of the SDC output directory in which to store the SDC constraints.
- Interblock logic – Select this option to perform top-level fixed delay budgeting. It is expected that the top-level delay will be optimized after the budgets are generated. The budgeted delay for the top-level path is exactly the same as the current delay at the top-level path segment. The default is on.

When this option is deselected, top-level proportional budgeting is performed. The input and output delays for the ports on soft macros or plan groups are allocated, considering the top-level delay as proportional.

- Advanced budgeting based on Virtual IPO – Select this option to perform a virtual in-place optimization before allocating slack to predict possible timing improvements. The virtual in-place optimization imitates the physical optimization to improve timing by virtually performing driver sizing for the cells on critical nets by up-sizing gates to their highest drive strengths or by inserting repeaters on critical nets to fix timing and DRC violations. The improved timing can expose blocks that require additional synthesis. This virtual in-place optimization does not perform any physical changes to the design; it is just held in memory for use with the budgeter and timing analyzer when what-if analysis is performed. The default is off.

Timing budgets are based on the estimated optimization.

- Incremental budgeting – Select this option to perform budgeting in incremental mode. During incremental budgeting, the timing budgeter retrieves block implementation information from designs with interface logic models (ILMs) and budgets the delays using this information. The block-level slack of the worst timing paths going through the block pins is taken into account. Pins associated with positive slack numbers will have their budgets tightened, and pins with negative slack numbers will have their budgets relaxed.

Note:

Use this option only on designs with interface logic models that were created using the `create_ilm` or `create_ilm_model` commands. Designs created by these commands contain information about the block-level slack of the worst path through the hierarchical pins. If the design contains plan groups or if the ILMs in the design do not have pins with negative slack information, incremental budgeting will issue an error message and stop.

- Split long lines – Select this option if you want long SDC command lines in a budget file split into multiple lines by using the “\” Tcl continuation character at the end of the line. The default is on.
- Enable QTM model generation – Select this option to enable quick timing model generation for the budgeted plan groups or soft macros. (This option is equivalent to the `create_qtm_model` command.) The delay arcs and the constraint arcs that are created in the quick timing models will represent the budgets on the plan groups or soft macros.

You can load the quick timing models for the soft macros or plan groups after they are committed to soft macros. You can also generate a top-level timing report to check the QoR of your budgets and identify potential problems before running optimization on the individual blocks.

- Output QTM directory – Enter the name of the directory into which the quick timing model files are written after the timing budgeting is finished. The default is the current directory.

Checking the QoR of the Timing Budgets

After you have loaded the quick timing models into your design, you can check the QoR of the timing budgets before running optimization for each individual block by using the `report_timing` command to create a top-level timing report to see if there are any paths with negative slack.

If the current budgets are either overconstrained or underconstrained, the timing report will show such paths as negative or positive slack. To check the results for any negative slack paths in the timing report, run a timing report for that path on a saved design with ILM models.

Zero slack budgeted paths are reported in the timing report with a slack of zero. Because the quick timing represents the timing of the blocks if they met the generated SDC budgets, the top-level timing should converge to a slack of zero.

Generating a Quick Timing Model From a CEL View

You can generate quick timing models for the current design or the soft macro cells inside the current design. The quick timing models are created, based on the clock information provided in the given SDC files, and the delay information in the quick timing models is based on the real delays in the design.

To generate quick timing models,

1. Choose Timing > Generate QTM Model.

The Generated QTM Model dialog box appears.

Alternatively, you can use the `generate_qtm_model` command.

2. Set the options, depending on your requirements.

- Soft macro cells or current design – Select the soft macro cells, and specify a list of soft macro names from which to create quick timing models. By default, a quick timing model is created for the current design.
- Cell name or SDC file name – Specify the list of cell names for the soft macros in the current design for which you want to create quick timing models.

Specify the SDC file(s) which contain clock information for the design or the soft macros. When creating quick timing models for the soft macros inside the current design, use this option together with the “soft macro cells” option and specify a list of cell names and SDC file names with this option to provide one SDC file for each soft macro cell.

- Output QTM directory – Specify the directory to where the quick timing model files will be written. By default, the files are written to the current working directory.

3. Click OK or Apply.

Generating a Quick Timing Model for the Partitions of Large Designs

Large designs that may not fit as comfortably as flat designs in IC Compiler are automatically partitioned and floorplanned. You can create a quick timing model for the partitions of large designs. The input to the quick timing model that is generated is the CEL

view of the partition. The quick timing model, along with the top-level design, should fit comfortably in a design that is floorplanned in IC Compiler. The quick timing model timing generation for all CEL view partitions can occur in parallel.

The generated quick timing model output for the partitions of large designs will have

- Interface ports of the output quick timing model.

The interface ports of the output quick timing model are based on the ports of the original CEL view. The direction of the quick timing model ports is the same as the direction of the ports in the Milkyway design.

- Sequential delay arcs (clock to output delay arcs)

If the worst slack path through the output port does not start at the input port of the design, a sequential delay arc is created at the output port.

- Combinational delay arcs (input to output delay arcs)

If the worst slack path through the output port starts at the input port of the design, a combinational delay arc is created.

- Setup and hold arcs (clock to input constraint arcs)

The worst slack path for maximum and minimum delay mode are used to create setup and hold arcs.

- Load information on input pins and drive information on output pins

The load value on the input port is the sum of all the pin and wire capacitances that are connected to the input port. The driving cell of the output port is the cell that directly drives the net connected to the output port.

Quick Timing Model Command Summary

[Table 12-1](#) summarizes the commands in the quick timing model flow.

Table 12-1 IC Compiler Quick Timing Model Command Summary

To do this	Use this command
Begin defining a quick timing model.	<code>create_qtm_model</code>
Create a constraint arc.	<code>create_qtm_constraint_arc</code>
Create a delay arc for a quick timing model.	<code>create_qtm_delay_arc</code>
Create a drive type in a quick timing model description.	<code>create_qtm_drive_type</code>

Table 12-1 IC Compiler Quick Timing Model Command Summary(Continued)

To do this	Use this command
Create a generated clock for the model.	<code>create_qtm_generated_clock</code>
Create a load type for a quick timing model description.	<code>create_qtm_load_type</code>
Create a path type in the quick timing model.	<code>create_qtm_path_type</code>
Create a quick timing model clock.	<code>create_qtm_clock</code>
Create a quick timing model port.	<code>create_qtm_port</code>
Create a quick timing model from a CEL view.	<code>generate_qtm_model</code>
Report model data.	<code>report_qtm_model</code>
Save the quick timing model.	<code>save_qtm_model</code>
Set a global parameter.	<code>set_qtm_global_parameter</code>
Set drive on a port.	<code>set_qtm_port_drive</code>
Set load on ports.	<code>set_qtm_port_load</code>
Set various technology parameters.	<code>set_qtm_technology</code>
Write out the quick timing model.	<code>write_qtm_model</code>

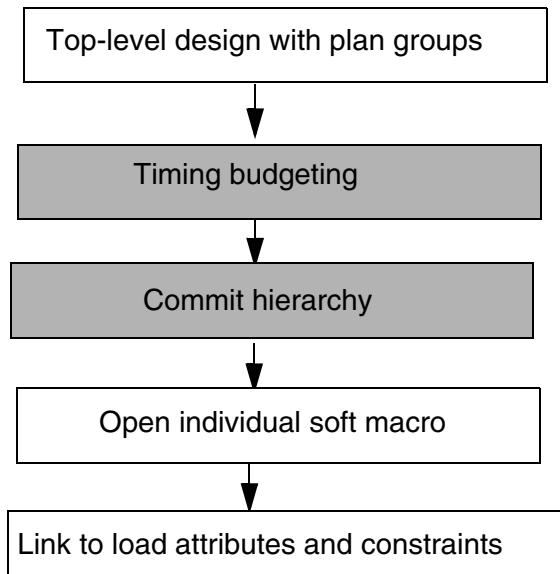
Performing Timing Budgeting On Plan Groups

If you have a design with plan groups, the top-level design will contain the complete design netlist. The `allocate_fp_budgets` command generates the SDC constraints and attributes for all the individual plan groups. These constraints and attributes are then transferred to the individual soft macro blocks when the hierarchy is committed. See “[Converting Plan Groups to Soft Macros](#)” in Chapter 13.

Note:

The automatic transfer of budgeted timing SDC constraints and attributes works on designs with single or multiple scenarios.

Timing budgeting on plan groups runs with full-chip timing. It honors the pin locations and the feedthrough nets assigned to the plan groups.



Note:

After committing the hierarchy using the `commit_fp_plan_groups` command, the created soft macros do not have timing models. At this point, you might encounter a few warning messages. These messages will not affect any physical design operations, such as refining the pin assignment.

If you want to perform an early timing check at the top level after you commit the hierarchy, do the following:

1. Save the soft macro CEL views by using the `save_mw_cel -hier` command.
2. Close the soft macro CEL views by using the `close_mw_cel` command, which leaves open only the CEL view for the top-level design.
3. Generate interface logic models (ILMs) for the soft macros from the top level by using the `create_ilm_models` command.
4. Run a top-level timing report using the ILMs.

This top-level timing report uses virtual-route based timing only, as there is no global route or parasitic information for the ILMs.

Performing Hierarchical Signal Integrity Budgeting

Timing budgeting can consider crosstalk effects across soft macro boundaries when generating SDC constraints. Hierarchical signal integrity budgeting can extract effective drive strength for input pins and coupling capacitance for interface nets. The timing budgeter can retrieve crosstalk analysis information from a hierarchical signal integrity module and then write out this information to block pin attributes. Bringing the top-level crosstalk information into the block level during budgeting makes it easier for block-level crosstalk analysis to consider the crosstalk effects at the top level.

Before running timing budgeting, set the `enable_hier_si` variable to true to enable signal integrity budgeting and take the crosstalk timing effects across soft macro boundaries into account.

```
icc_shell> set enable_hier_si true
```

Run the timing budgeting command.

```
icc_shell> allocate_fp_budgets
```

IC Compiler will process the hierarchical signal integrity information during budgeting and will write out this information on the block's pin attributes.

The total effective drive strength is written out on block pins to represent the top-level aggressor drive strength, and the total coupling capacitance is written out on block pins to represent the total coupling capacitance in the block CEL view.

Block-Level Hierarchical Signal Integrity Flow

To run the block-level hierarchical signal integrity flow, do the following:

1. Enable the hierarchical signal integrity flow.

```
set enable_hier_si true
```

2. Define the signal integrity options used for analysis or optimization.

```
set_si_options -static_noise true -delta_delay true
```

The `-static_noise true` option reduces static noise.

The `-delta_delay true` option optimizes the timing with crosstalk delay.

3. Perform routing and postroute optimization.

```
route_opt -xtalk_reduction
```

IC compiler performs crosstalk prevention during global routing and track assignment, and crosstalk analysis and optimization during the postroute optimization phase.

4. Create a Milkyway signal-integrity-aware interface logic model (ILM).

```
create_ilm -include_xtalk
```

The `-include_xtalk` option includes crosstalk information, such as boundary net aggressor data, effective net resistance, total capacitance, and coupling capacitance.

Top-Level Hierarchical Signal Integrity Flow

To run the top-level hierarchical signal integrity flow, do the following:

1. Enable the hierarchical signal integrity flow.

```
set enable_hier_si true
```

2. Define the signal integrity options used for analysis or optimization.

```
set_si_options -static_noise true -delta_delay true
```

The `-static_noise true` option reduces static noise.

The `-delta_delay true` option optimizes the timing with crosstalk delay.

3. Perform routing and postroute optimization.

```
route_opt -xtalk_reduction
```

IC Compiler performs crosstalk prevention during global routing and track assignment, and crosstalk analysis and optimization during the postroute optimization phase.

Performing Post-Budget Timing Analysis

Immediately after budgeting a design, you can use the `check_fp_budget_result` command to perform post-budget analysis. It generates a report containing budgeted and actual delays through a hierarchical block.

Note:

This command must be run during the same IC Compiler timing budgeting session in which the budgets were created.

A timing path starts at a startpoint and ends at an endpoint. For each timing path segment (timing paths are divided into timing segments by hierarchical pins), the report prints the actual and budgeted delay or the fixed delay value of the segment. You can use this information to analyze the process of generating budgets for your design.

For each pin on a block, the report lists the worst-case timing paths per endpoint clock domain that go through the pin. For each path segment on the timing path, the report lists the budgeted and actual delays for the path segment.

The `check_fp_budget_result` command uses the following syntax:

```
check_fp_budget_result
[-blocks block]
[-pins pin]
-file_name output_design_report
```

Note:

If no option is specified, information for the whole design is written out.

Option	Description
<code>-blocks <i>block</i></code>	Specifies the name of a plan group in the design. It reports the timing analysis information related to that particular plan group in the design. You can specify the value of <i>block</i> by using a collection generated by the <code>get_cells</code> command. By default, a budgeting analysis report is generated for all the plan groups in the design.
<code>-pins <i>pin</i></code>	Specifies the name of a pin on a plan group in the design. It reports the timing analysis information related to that particular pin in the design. You can specify the value of <i>pin</i> by using a collection generated by the <code>get_pins</code> command. By default, a budgeting analysis report is generated for all the pins in all the plan groups.
<code>-file_name <i>output_design_report</i></code>	Specifies the name of the output design report file. The design information is written to this file. This option is mandatory.

Performing Clock Latency Budgeting

You can perform clock latency budgeting. This budgeting feature uses information from files built by clock planning to include clock latencies associated with real clock tree branches that are external to the block being budgeted. Clock planning can create clock budgets for each plan group inside the design to specify the source and network latencies whether or not the clock has a sink inside the plan group.

The `allocate_fp_budgets` command creates virtual clocks to capture these latencies from clock planning. This budgeting feature also generates real latency values in budgets for physical clock ports created by clock planning on a block. The clock ports are created by the clock planning features. These latencies are associated with real physical ports.

Creating Budgets to Reflect Real Clock Latencies

After clock planning, you can create budgets to reflect real clock latencies. This consists of:

- Creating real clock latency values on clock ports for the block.
- Creating input and output virtual clocks for interface path endpoints that are partially outside the block.

The format for defining virtual clocks created for latencies outside the block is

`clock_name_v_in`

and

`clock_name_v_out`

Clocks launching (capturing) flip-flops on the top level have source latencies only.

Clocks launching (capturing) flip-flops in blocks have both source and network latencies.

Clock latency budgeting is on by default if timing budgeting (`allocate_fp_budgets` command) is run after clock planning.

You can enable (turn on) clock planning based latency values by setting the following variable in the `icc_shell` to `true` (the default) before performing timing budgeting:

```
set virtual_clocks_from_cp true
```

To disable (turn off) clock planning based latency values, set the following variable in the `icc_shell` to `false`.

```
set virtual_clocks_from_cp false
```

For a design which has synthesized clock trees in it which were not created by clock planning, you can have latencies for these existing clock trees included in budgeting by setting the following variable to a value of `true` before running the timing budgeter. The default is `false`.

```
set synthesized_clocks true
```

To disable (turn off) clock tree synthesis based latency values, set the following variable in the `icc_shell` to `false`.

```
set synthesized_clocks false
```

Things to Look For

After you create budgets to reflect real clock latencies, check for the following:

- Ensure every real input clock port created by clock planning for a block budget has calculated source and network latency statements in the block budget.
- Ensure latency statements in the budgets reflect the actual clock network delays in the design.
- Ensure every input port `set_input_delay` and `set_output_delay` is referenced to a virtual clock that has the latency for the launch or capture clock associated with the path through the port.
- Check latencies in budgets against clock planning latency files.

Note:

Clock planning creates a source latency file and a network latency file for each clock port in the `tcp_output` directory.

- Check that latencies in budgets are only for clock ports defined in the block budget file.
- Check latencies in clock planning latency files against the clock skew report.

13

Committing the Physical Hierarchy

This chapter describes how to commit the physical hierarchy after finalizing the floorplan by converting plan groups into soft macros. Committing the hierarchy creates a new level of physical hierarchy in the virtual flat design by creating CEL views for selected plan groups. After committing the physical hierarchy, you can also “uncommit” the physical hierarchy by converting the soft macros back into plan groups.

In addition, this section also describes how to propagate top-level preroutes into soft macros, recover all pushed-down objects in child cells to the top-level, and uncommit the physical hierarchy by converting soft macros back into plan groups. It also describes how to commit plan groups with associated UPF power domains.

This chapter includes the following sections:

- [Converting Plan Groups to Soft Macros](#)
- [Pushing Physical Objects Down to the Soft Macro Level](#)
- [Pushing Physical Objects Up to the Top Level](#)
- [Handling 45 Degree Redistribution Layer \(RDL\) Routing](#)
- [Committing the Hierarchy of Plan Groups With Unified Power Format \(UPF\) Power Domains](#)
- [Converting Soft Macros to Plan Groups](#)
- [Propagating United Power Format \(UPF\) Constraints From the Soft Macro to the Top Cell](#)

Converting Plan Groups to Soft Macros

You can convert selected plan groups or all plan groups with exclusive placement constraints into soft macro CEL views. The soft macros are created in the CEL view directory.

Prerequisites

Before converting plan groups to soft macros, your design should meet the following requirements:

- Exclusive plan groups, standard cells, and hard macros are legally placed.
- Traditional pin assignment or pin cutting flow is completed.
- In-place optimization, timing analysis, and timing budgeting are completed.

To convert plan groups to soft macros,

1. Choose Partition > Commit Plan Groups.

The Commit Plan Groups dialog box appears.

Alternatively, you can use the `commit_fp_plan_groups` command.

2. Set the options, depending on your requirements.

- Specify whether to commit all plan groups or specified plan groups. The default is all.
- Push down power and ground straps into the child cell – Select the option if you want power and ground straps copied (pushed) down into the newly created soft macro CEL views. The power and ground straps are removed from the top level. The default is off.
- Commit to a new top cell – Select this option to commit the plan groups to a new top cell. A new top cell is created in your design where the top level changes occur. The default is off.
- Top cell name – Enter the name of the new top-level cell.

3. Click Apply or OK.

Committing the physical hierarchy does the following:

- Converts the virtual flat design to a two-level hierarchical design.
- Removes standard cells and hard macros belonging to a plan group from the top cell and copies them down to the soft macro CEL which is created for the plan group.
- Creates nets in the soft macro CEL views based on the Hierarchy Preservation data.
- Pushes placement blockages and route guides down to the soft macro CEL views.

- (Optional) Pushes power and ground straps and standard cell preroutes down into each soft macro CEL view.

After the conversion from plan groups to soft macros is complete, all the standard cell instances of the plan groups become child cell instances of the corresponding converted soft macro cells and are deleted from the top cell. New corresponding cell instances of the new soft macro cells are created in the top cell and reside in the same location as the old plan groups.

Utilization of the new CEL views is the same as the utilization of the corresponding plan groups prior to running the `commit_fp_plan_groups` command. The default utilization is based on the sum of the area of standard cells and hard macros in the CEL view divided by the area in the core area bounding box of the CEL.

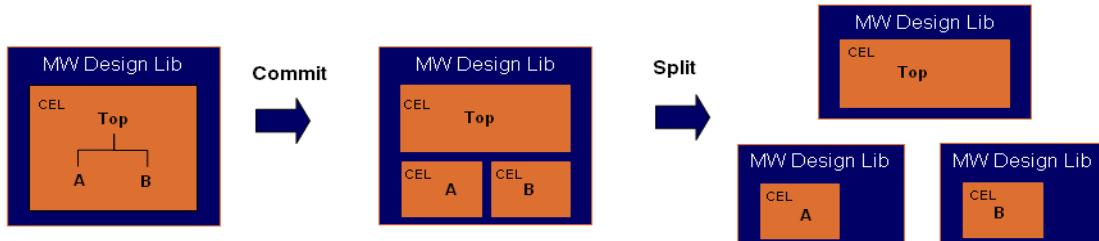
Splitting the Soft Macros

When you commit the soft macros, the IC Compiler tool creates a CEL view for each of the blocks in the same Milkyway library. To allow different designers to work on different blocks without interference, you can use the `split_mw_library` command to split the top-level Milkyway design into new Milkyway design libraries for each block and the top-level design.

After splitting, the new Milkyway design libraries inherit all the information, including the technology file and the reference library path, from the original Milkyway design.

[Figure 13-1](#) shows the splitting of soft macros to create separate design libraries.

Figure 13-1 Splitting of Soft Macros



After block-level implementation, you can link the CEL views for the blocks in different design libraries to the top-level design by using the `set_mw_lib_reference` command.

Pushing Physical Objects Down to the Soft Macro Level

Physical objects (routing, route guides, blockages, cells, and cell rows) can be transferred (pushed down) from the top level to the soft macro level.

To push physical objects down to the soft macro level,

1. Choose Partition > Push Down Objects.

The Push Down Objects dialog box appears.

Alternatively, you can use the `push_down_fp_objects` command.

2. Enable the “Push down objects” option. This is the default.

3. Set the remaining options, depending on your requirements.

- Target soft macros – Select “All soft macros” (the default) or “Specified soft macros”. If you select the “Specified soft macros” option, enter the names of the soft macros and optionally specify the objects to be pushed down into the specified macros.

- Types of objects to be pushed down – Select the types of objects to be pushed down. The default is Routing.

Routing – This object type includes wires, paths, and vias. If a net is connected to these objects at the top level, but it does not exist in the soft macro, the net is created inside the soft macro.

Route guides – This object type includes route guides and their corresponding attributes.

Blockages – This object type includes metal layer blockages and soft or hard placement blockages.

Cells – This object type includes all standard cells within the boundary of the soft macros marked as flip-chip area I/O cells or drivers. If a net is connected to these cells at the top level, but it does not exist in the soft macro, the net is created inside the soft macro.

Rows – This object type includes all cell rows that touch the specified soft macros.

- Net type – You can specify the types of nets that are processed. You can specify more than one type of net.

Power and ground – Select this option to process power and ground nets. This is the default. When power and ground nets are pushed down, the associated via cell instances are pushed down as well.

A power and ground wire or path passing through the soft macro area is removed from the parent cell and moved into the child cell. If the power and ground net is connected to an existing port, new pins are created. If the power and ground net is not connected to an existing port, this wire, path, or via is not pushed down.

Clock – Select this option to process clock wires.

Clock wires or paths passing through the soft macro area are removed from the parent cell and moved into the child cell. If the clock net is connected to an existing port, new pins are created. If the clock net is not connected to an existing port, one new port and new pins are created.

Signal – Select this option to process non-bus signal wires or paths.

Non-bus signal wires or paths passing through the soft macro area are removed from the parent cell and moved into the child cell. If the signal net is connected to an existing port, new pins are created. If the signal net is not connected to an existing port, one new port and new pins are created.

- **Cut type** – You can specify how the tool should process the pushed-down objects.
 - Push down – Creates a copy of the object in the soft macro and then deletes the object from the top level. This method applies to routing, route guides, blockages, and cell object types. This is the default.
 - Cut down – Creates pins for routing objects, where the routing object touches the soft macro boundary and then deletes the object from the top level.
 - Copy down – Creates a copy of the object in the soft macro, but keeps the object in the top level.
- **Don't check overlapping** – If you select this option, physical object types are pushed down without checking for routing overlaps. For example, if objects overlap on the same metal layer with a wire, path, via, or a pin with a different net type, they are ignored, even if they cause shorts. The default is off.
- **Connect copy down wires** – If you select this option, wires are copied down into the soft macro and assigned (connected) to the parent net ID. The default is off.
- **Freeze push down nets** – If you select this option, the routing constraints contained in the pushed down nets are frozen. The default is off.
- **Allow feedthroughs** – If you select this option, the command creates new ports for a feedthrough route and creates a new net at the top level for the split net. Otherwise it does not create new ports; if the top-level net has a connection with the soft macro, it creates pins at each crossing point.
- **Allow partially overlap soft macros** – If you select this option, cells which only partially overlap the soft macro boundary are pushed down.

- Horizontal offset and Vertical offset – You can specify the horizontal offset and vertical offset between the soft macro core and the cell rows. The default is 0.
4. Click OK or Apply.

Pushing Physical Objects Up to the Top Level

Physical objects that were previously pushed down into a soft macro can be pushed back up to the top level.

To push up physical objects from the soft macros in the design to the top level,

1. Choose Partition > Push Up Objects.

The Push Up Objects dialog box appears.

Alternatively, you can use the `push_up_fp_objects` command.

2. Enable the “Push up objects” option. This is the default.

3. Set the remaining options, depending on your requirements.

- Target soft macros – Select “All soft macros” (the default) or “Specified soft macros”. If you select the “Specified soft macros” option, enter the names of the soft macros from which to push up objects. By default, objects are pushed up from all soft macros in the current design.
- Types of objects to be pushed up – Select the types of objects (routing, route guides, pins, blockages, cells, and rows) to be pushed up to the top level. The default is Routing.

Routing – This object type includes wires, paths, and vias.

Cells – This object type includes all standard cells within the boundary of the specified soft macros.

Route guides – This object type includes route guides and their corresponding attributes.

Rows – This object type includes all cell rows that touch the specified soft macros. The rows are not pushed up to the top-level because the top level already contains the row objects; the rows are just removed from the soft macros.

Blockages – This object type includes all blockages on metal blockage layers, and soft or hard placement blockages.

Pins – This object type includes the hard and soft macros’ pins.

- Preserve child preroutes – If you select this option, child preroutes are preserved. The top-level preroutes are re-created in the parent directory without deleting child-level objects from the soft macro. The default is off.

- Types of objects after push up from soft macros – Use this option with pushed up soft or hard macro pins (select the object type “Pins”, which includes top-level pins”) or with routing (select the object type “Routing”, which includes wires, paths, and vias”) only. You can specify the top-level object types after they are pushed up from the soft macro level. You can push up child level routing to top-level routing or to top-level pins if there are top-level ports on connected top-level nets. If the child level routing is pushed up to top-level pins and the routes are rectilinear, the pushed up routes are broken into rectangular EEQ pins.

Alternatively, you can push up soft or hard macro pins from the soft macro level back up to the top level as wire routing or as top-level pins.

- Layers of routings/pin objects – If you select this option, you can limit the objects that are pushed up to specific layers. You should use this option with routing and pin objects only.
- Top level nets connected to soft macros – If you select this option, you can specify a list of top-level interface nets for which routing or pin object types will be pushed up. Only child objects with logical connections to those top-level nets are pushed up to the top level.
- Pushed down objects only – If you select this option, you can define the objects to be pushed up to the top level. It applies only to cells, nets, and routing objects. By default, only the object types that were previously created during push down are pushed up to the top level. If you select “all”, the object types that are created solely inside soft macros are also pushed up to the top level.

Push up ignores this option for route guides, blockages, and row object types and will push up both previously pushed down objects and objects created solely inside the soft macros.

- Hierarchical names of child objects – If you select this option, you can selectively push up objects inside soft macros to the top level by entering a full hierarchical cell instance name or net name.

4. Click OK or Apply.

Handling 45 Degree Redistribution Layer (RDL) Routing

The `push_up_fp_objects` and `push_down_fp_objects` commands can support 45 degree redistribution layer (RDL) routing. During push-down, the 45 degree routings are split at the intersecting points between the soft macro boundaries and where the 45 degree routings are added. Square-shaped pins are created at the intersection points during push-down; during push-up, these pins are deleted at the soft macro boundaries.

During push-up, the 45 degree routings are restored to the original state prior to push-down. (Wires are merged at the intersection points between the soft macro boundaries and routings.)

Committing the Hierarchy of Plan Groups With Unified Power Format (UPF) Power Domains

The `commit_fp_plan_groups` command supports plan groups which have a unified power format (UPF) power domain associated with them. The power domain must be defined completely within the hierarchical cell instance of the plan group that is being converted to a soft macro. The `create_power_domain` command defines a power supply distribution network at the current scope (hierarchical level) or at the scope of a specified hierarchical instance. The `-scope` option specifies the name of an instance in which to create the power domain; it defines the domain boundary within the logic design. In a hierarchical UPF context, a “scope” is a hierarchical cell instance within which the power domain is completely contained.

During the commit hierarchy process, the top-level UPF power domains associated with the plan groups are pushed down into a new child cell view. This also makes it possible for the top-level design’s UPF constraints to be properly transferred and maintained. Embedded voltage areas and power domains associated with those voltage areas are also pushed into the child cell.

Near the end of the commit process, after the top-level hierarchical cell instance is replaced with a top-level child cell instance, all corresponding power domains are removed from the top-level cell, along with any voltage areas associated with the pushed down power domains. This is done to avoid the duplication of power domains across the hierarchy. Any top-level voltage area which was coincident with the plan group boundary, is also removed from the top cell and treated as `DEFAULT_VA` within the child cell instance.

Note:

If UPF power domains are not completely defined within the hierarchical cell instance, in other words they are larger than the plan groups associated with them, it means they might contain UPF elements that are outside the plan group and therefore, they cannot be committed.

Converting Soft Macros to Plan Groups

After plan groups are committed into soft macros, you can “uncommit” the physical hierarchy by converting the soft macros back into plan groups. The physical hierarchy is flattened into the parent.

Physical objects from the soft macro are “pushed up” to their relative positions in the top CEL view. Any physical objects that are connected to power and ground nets will be connected to their appropriate nets in the top CEL view, and any objects that have properties set in the soft macro will have the same properties set in the top CEL view.

To uncommit the physical hierarchy,

1. Choose Partition > Uncommit Soft Macros.

The Uncommit Soft Macros dialog box appears.

Alternatively, you can use the `uncommit_fp_soft_macros` command.

2. Choose whether to convert all soft macros or selected soft macros to plan groups.

3. Set the remaining options, depending on your requirements.

- Preroutes to push up – Select the “All” option (the default) to push up all routing, including both power and ground, and detail routing. Select the “Power and Ground” option if you want only power and ground straps pushed up to the top CEL view.
- Remove feedthrough ports on soft macros – Select this option to remove the feedthroughs from selected soft macros. All feedthrough nets and ports with name suffixes of `*_pft` and `*_rft` are removed.

4. Click OK or Apply.

When you click the OK or Apply button in the dialog box, the following physical objects are pushed up to the top CEL view from soft macros:

- Standard cells
- Straps on power and ground nets
- Vias on power and ground nets
- Placement blockages, including soft placement blockages and the internal padding around the boundary and cell rows
- Route guides

Keep the following points in mind when you uncommit the hierarchy:

- The uncommit process does not remove the soft macros that were created in the CEL views.
- Any routing done on the soft macros is deleted after you uncommit the hierarchy (except for the power and ground preroutes).

Propagating United Power Format (UPF) Constraints From the Soft Macro to the Top Cell

During the uncommit process when the soft macros are converted back into plan groups, the `uncommit_fp_soft_macros` command automatically propagates the child level UPF constraints and the Milkyway and Design Compiler information from the soft macro to the top level of the design.

During the uncommit hierarchy process, the `uncommit_fp_soft_macros` command causes the following actions when it propagates the UPF constraints to the top level.

- Locates all UPF power domains and their corresponding voltage area geometries and instantiates them in the top cell, according to how they existed in the soft macro. In the case of the top-most UPF power domain in the soft macro, which has a `DEFAULT_VA`, the voltage area is instantiated in the top cell with the same name as its corresponding power domain and with a geometry that is coincident with the plan group boundary.
- Reconnects the newly instantiated top-cell UPF Milkyway power domains to the existing UPF Milkyway objects.
- Transfers the Design Compiler UPF constraints from the soft macro NID cell attach files to the top-cell level NID plan group attach file.

A

Using the Flip-Chip Flow

This appendix describes how to use the flip-chip flow in IC Compiler. You can use hierarchical or virtual flat implementation flows for designs with flip-chip structures. The IC Compiler flip-chip interface can create both 45 degree and 90 degree redistribution layer routing.

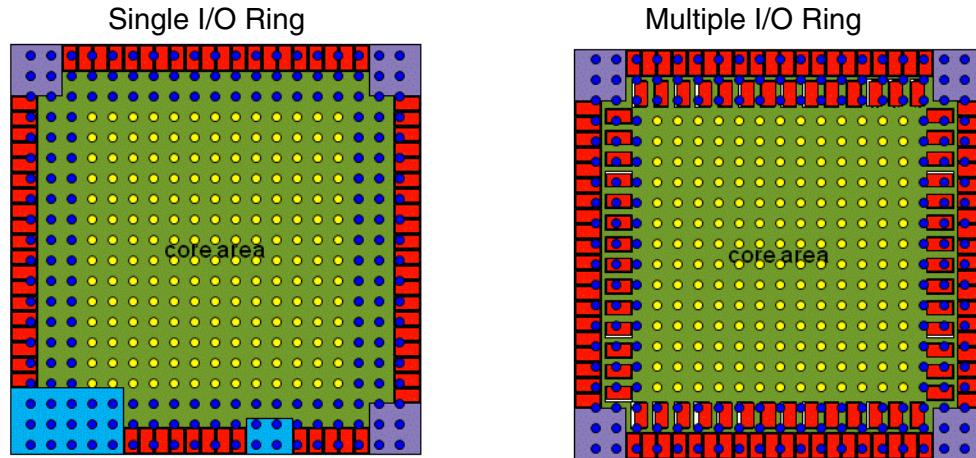
The following two flip-chip design flows are supported in IC Compiler design.

- Package-driven, also referred to as a bump-driven flow, where flip-chip I/O driver locations are optimally determined by package defined bump cell locations
- Die-driven, also referred to as an IC-driven flow, where individual bump cell locations are optimally determined by the I/O driver placement

The IC Compiler flip-chip interface supports two design styles: the single-ring perimeter I/O driver and the multiple ring perimeter I/O driver. In a perimeter ring I/O style, the flip-chip drivers are placed in a ring formation along the periphery of the chip.

[Figure A-1 on page A-2](#) shows a peripheral single-ring I/O style and a multiple ring I/O style.

Figure A-1 Single-Ring and Multi-Ring I/O Styles



In a flip-chip design, a bump is a special cell representing a piece of metal, also called a solder bump, that forms an electrical contact to the chip's packaging. Similar to processing silicon wafers for wire bonding, openings are made on the wafer to expose the metal contact points. Solder bumps are formed on these metal contact points. The chip's power, ground, and signal I/Os are available through the solder bumps that are formed on the top side of the wafer during the final processing step. The bumps on the chip make the physical connections to the package substrate. The chip is connected to external circuitry (a circuit board, another chip, or wafer) by "flipping" it over so that the solder bumps connect to metal pads to complete the interconnect. This is in contrast to wire bonding in which the chip is mounted upright and wires are used to connect the chip pads to external circuitry.

The flip-chip driver is the I/O circuitry that drives or receives signals from or to the chip through the bump. There are signal bump cells, which are connected to signal nets, and power and ground bump cells, which are connected to power and ground straps. The locations of bumps on the chip are determined by the placement of the bump cells. (The I/O driver cells are placed independently.) You can place bump cells on top of the I/O drivers or anywhere within the core area. You can also place standard cells under bump cells

This appendix includes the following sections:

- [Preparing the Library](#)
- [Creating an Initial Die Size](#)
- [Using a Die-Driven \(IC-Driven\) Driver and Bump Placement Flow](#)
- [Using a Package-Driven \(Bump-Driven\) Bump and Driver Placement Flow](#)
- [Performing Automatic Bump Net Routing](#)
- [Using Flip-Chip Structures in Cover Macros](#)

- [Summary of the Flip-Chip Commands](#)

Preparing the Library

IC Compiler recognizes flip-chip I/O drivers and bump cells only when they are marked as flip-chip type. This section describes the library preparation for the flip-chip I/O drivers and bump cells.

I/O Drivers

To prepare the I/O drivers, do the following:

- Set the cell type as "flip chip driver" by using the `cmMarkCellType` command.

```
cmMarkCellType
setFormField mark_cell_type library_name library name
setFormField mark_cell_type cell_name cell name
setFormField mark_cell_type cell_type "flip chip driver"
formOK mark_cell_type
```

- Set the PAD port as "flipchip" by using the `dbSetCellPortTypes` command.

```
dbSetCellPortTypes
("PAD" "inout" "flipchip")
```

- Set the power and ground pins as flipchip "power" and "ground" by using the `dbSetCellPortTypes` command.

```
dbSetCellPortTypes
("VDD" "inout" "Power")
("VSS" "inout" "Ground")
```

The power and ground I/O drivers that connect to power and ground bumps should also have a "flipchip" property defined for the pins.

```
dbSetCellPortTypes
("VDD" "inout" "Power" "flipchip")
("VSS" "inout" "Ground")
```

Bump Cells

To prepare the bump cells, do the following:

- Mark the bump cell as a flip-chip bump by using the `cmMarkCellType` command.

```

cmMarkCellType
setFormField mark_cell_type library_name library name
setFormField mark_cell_type cell_name cell name
setFormField mark_cell_type cell_type "flip chip pad(bump)"
formOK mark_cell_type

```

You need signal bumps and power and ground bumps.

Signal bumps are used to connect signals to I/O drivers.

Power and ground bumps are arrayed over the core area with minimum spacing for power and ground delivery by using the `place_flip_chip_array` command.

For example:

```

place_flip_chip_array
-physical_lib_cell{BUMP100_P}
-prefix "VDD_CORE_"
-start_point {1055 1090}
-number 555
-delta {270 540}
-repeat {27 13}

```

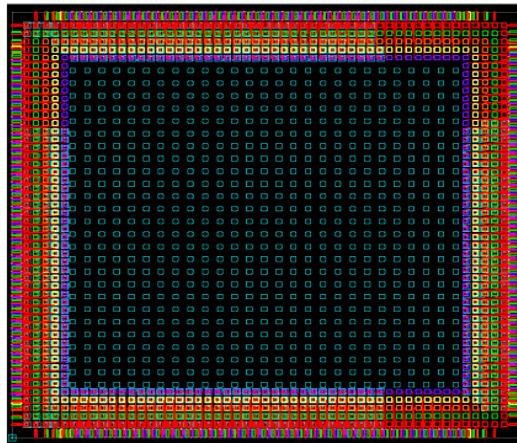
```

place_flip_chip_array
-physical_lib_cell{BUMP100_G}
-prefix "VSS_CORE_"
-start_point {1055 1360.426}
-number 555
-delta {270 540}
-repeat {27 13}

```

[Figure A-2](#) shows the power and ground bumps arrayed over the core area.

Figure A-2 Power and Ground Bumps Arrayed Over Core Area



- Set the port as "flipchip" by using the `dbSetCellPortTypes` command.

- Create bump terminals (pins) on a redistribution routing layer.
To create a redistribution layer routing terminal,
 - Create .lib file with a terminal name and a corresponding .db file so that IC Compiler can create the bump cells and perform the logical connectivity. Without the .lib file, all bump cells are treated as physical-only cells.
 - Locate the layer for the octagonal-shaped passivation opening in the flip-chip bump cell.
 - Create a polygon in the CEL view by using the `geAddPolygon` command or choose Milkyway: Create > Polygon. Query the polygon to get a list of vertices.
 - Enter the name or number of the layer on which you want to create the polygon, select the L45 routing option, and then add all the vertices.

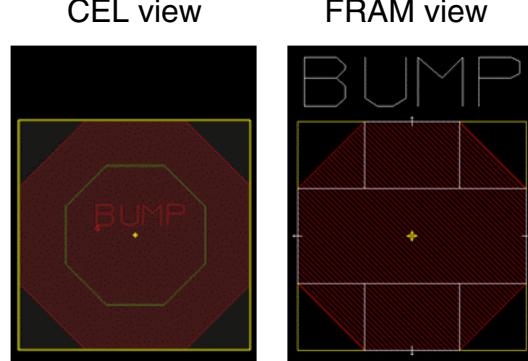
You should now have an octagonal piece of metal in the bump CEL view.

 - Add text to the layer by using the `geAddText` command. Use the text layer that matches the newly created terminal.
 - Create a FRAM view by using the `auExtractBlockPinVia` command.

The CEL and FRAM view should match the terminal name that you created in the .lib file.

[Figure A-3](#) shows the CEL and FRAM views of a bump cell.

Figure A-3 Bump CEL and FRAM Views



Creating an Initial Die Size

Use the `initialize_floorplan` command to create an initial die size for the core area of your design, based on input control parameters such as target utilization, aspect ratio, core size, row number, chip boundary, and wire tracks.

Using a Die-Driven (IC-Driven) Driver and Bump Placement Flow

In a die-driven flow, the IC design dictates the flip-chip bump cell locations to the IC packages.

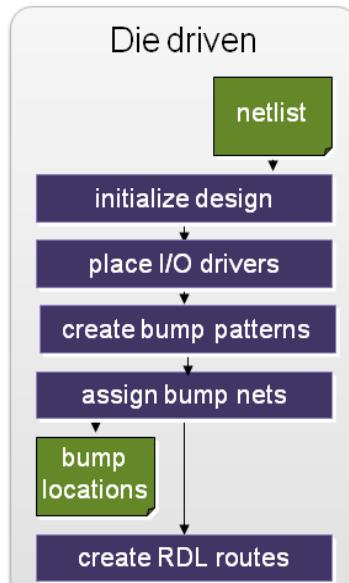
- The bump cells are not instantiated, nor are they connected to flip-chip I/O drivers in the Verilog netlist.
- Individual bump cell locations are optimally determined by the placement locations of the flip-chip I/O drivers and the bump pattern placement.
- The connectivity between the I/O ports and the I/O drivers can be output in the Verilog netlist.

The following topics are covered in this section:

- [Placing the Flip-Chip I/O Drivers](#)
- [Creating a Pattern of Bump Cells in a Ring Configuration](#)
- [Creating a Pattern of Bump Cells in an Array Configuration](#)
- [Automatically Assigning Nets From Bumps to I/O Drivers](#)
- [Assigning Bump Pattern Personality Types](#)

[Figure A-4](#) shows the die-driven design flow style.

Figure A-4 Die-Driven Design Flow Style



Placing the Flip-Chip I/O Drivers

In most cases, designers are provided with guidelines on where to place the flip-chip I/O drivers. For example, some I/O drivers are placed on the left-side of the chip in a particular order, other I/O drivers are placed along the top of the chip in a particular order, and so forth.

Top-level I/O pads are placed according to the constraints specified in the Top Design Format (TDF) file.

Note:

The TDF constraint file does not currently apply to I/O flip-chip drivers. Therefore, as an alternative, it is recommended that you maintain two Milkyway I/O libraries: one library should contain the regular I/O pads on which you can apply the TDF constraints when you initialize the die (`initialize_floorplan` command) to position the I/O pads following the placement guidelines; and the other library should contain the I/O drivers marked as "flipchip" by using the `cmMarkCellType` command.

Creating a Pattern of Bump Cells in a Ring Configuration

You can create a specified number of bump cells and place them in your design in a ring configuration by using the `place_flip_chip_ring` command. You can also specify the number of rings to be created and the spacing between adjacent rings.

The `place_flip_chip_ring` command uses the following syntax:

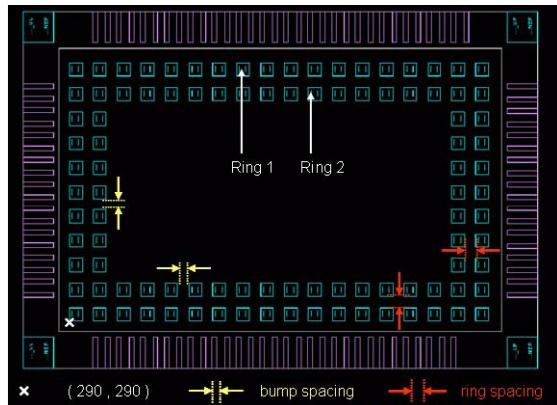
```
place_flip_chip_ring
  -physical_lib_cell cell_name
  -prefix prefix
  -number num_bump
  -bump_spacing spacing
  -ring_number ring_number
  -ring_spacing ring_spacing
  -boundary boundary_box
  [-left_orientation N | W | S | E | FN | FS | FW | FE]
  [-stagger_offset offset]
  [-extra_spacing extra_spacing]
  [-num_extra_spacing num_extra]
```

In the following example, 100 flip-chip bumps are created and placed in two rings. The spacing between the adjacent bump cells and the adjacent ring cells is 70 microns, and the outermost ring is set to the lower-left coordinates of (290 290) and the upper-right coordinates of (2940 1934).

```
place_flip_chip_ring -physical_lib_cell {PAD80B} -prefix "BUMP_"
  -bump_spacing 70 -ring_number 2 -ring_spacing 70 -number 100 -boundary
  "290 2940 1934"
```

[Figure A-5](#) shows the bump cells placed in a ring configuration.

Figure A-5 Placing Bump Cells in a Ring Configuration



Creating a Pattern of Bump Cells in an Array Configuration

You can create a specified number of flip-chip bump cells and place them in your design in a two-dimensional array pattern by using the `place_flip_chip_array` command. You specify the size of the array starting at a specified coordinate and the distance between adjacent bumps in the array pattern.

The `place_flip_chip_array` command uses the following syntax:

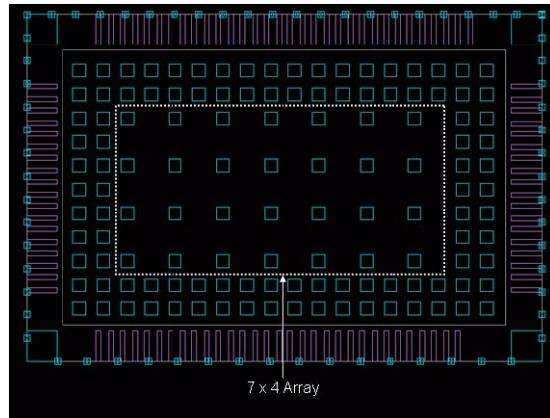
```
place_flip_chip_array
  -physical_lib_cell phys_lib_cels
  -prefix prefix
  -start_point start_point
  -number num_bump
  -delta {x y}
  -repeat {i j}
  [-orientation N | W | S | E | FN | FE | FS | FW]
  [-cell_origin lower_left | center]
```

In the following example, 28 flip-chip "VDD_CORE_*" bumps are created and placed in a 7 by 4 array pattern, starting at the lower-left x-coordinate of 590 and the lower-left y-coordinate of 590. There is an X-pitch and a Y-pitch of 300 microns between adjacent bump cells.

```
place_flip_chip_array -physical_lib_cell {PAD80B_P} -prefix "VDD_CORE_"
  -start_point {590 590} -number 28 -delta {300 300} -repeat {7 4}
```

[Figure A-6 on page A-9](#) shows the bump cells placed in a 7 by 4 array pattern.

Figure A-6 Placing Bump Cells in a 7 by 4 Array Pattern

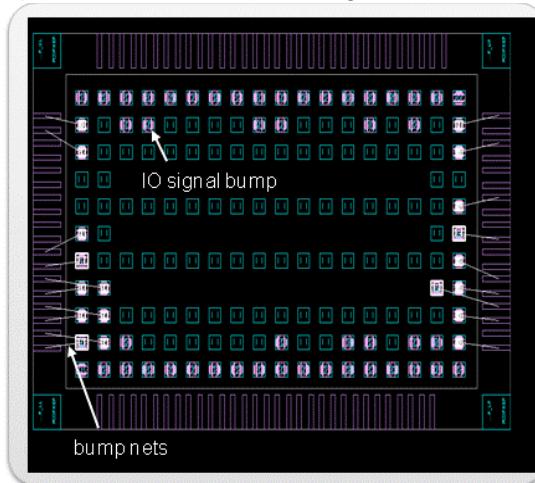


Automatically Assigning Nets From Bumps to I/O Drivers

Based on the current driver placement results, the `assign_flip_chip_nets` command locates the nearest unassigned flip-chip bumps, matches them to flip-chip I/O drivers using the shortest wire length method and then automatically assigns new logical nets or reconnects the nets between the flip-chip bumps and I/O drivers, as shown in [Figure A-7](#). The I/O drivers are placed considering the connectivity to the core logic. The bump cell instances are connected to nearby I/O drivers automatically.

When you run the `assign_flip_chip_nets` command, it also optimizes the bump-to-driver interconnect wire length.

Figure A-7 Automatic Bump Net and I/O Driver Assignment



The `assign_flip_chip_nets` command uses the following syntax:

```
assign_flip_chip_nets
[-personality_type personality_type_list]
[-prefix prefix]
[-uniquify num_to_uniquify]
[-eco]
```

Note:

The driver-to-bump matching is processed by personality types that were defined by the `set_flip_chip_type` command. Only drivers and bumps with identical personality types are matched.

For bumps with an existing logical connection, unless the `-eco` option is specified, the previous connection to the bump is removed, and a new bump net is assigned. The newly matched bump is reconnected to the matching flip-chip driver. The new bump net inherits the net name from the flip-chip driver pad to which the bump is now connected.

You can use the `-uniquify num_to_uniquify` option to specify whether to and how to uniquify the driver nets. This option takes effect only when there are multiple power drivers whose flip-chip ports are connected to the same net. The `num_to_uniquify` argument accepts integers from `-n` to `0` to `n`. The default is 1-to-1uniquify.

- `0` – Do not uniquify. Assign only one bump for the multiple flip-chip driver ports. The original driver net can be reused.
- `1` – 1-to-1 uniquify. Assign one bump for each flip-chip driver port. New nets are created with the original net name followed by `"_#"`. A VDD net, for example can be uiqified to VDD, VDD_1, VDD_2, and so forth.
- `n` – n-to-1 uniquify. Assign one bump for every `n` flip-chip driver port.
- `-n` – 1-to-n connection. Assign `n` bumps for each flip-chip driver port.

When the `-eco` option is specified, existing logical connections are kept, and a new logical net is created only if the flip-chip driver pad has no existing net connection.

Note:

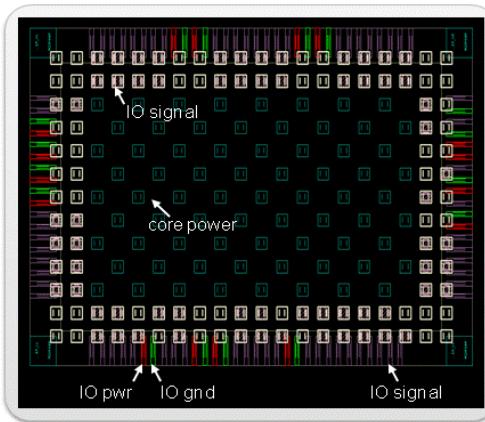
The `assign_flip_chip_nets` command can handle multiple PAD pins per I/O driver, with multiple bump cells connected to one single I/O driver with multiple pad pins.

You can also place I/O signal nets, I/O power and ground nets, and core power nets, as shown in [Figure A-8 on page A-11](#).

- The I/O signal drivers are placed considering the connectivity to the core logic.
- The I/O power and ground drivers are placed based on guidelines from the I/O provider.
- The signal bumps are placed close to the I/O drivers.

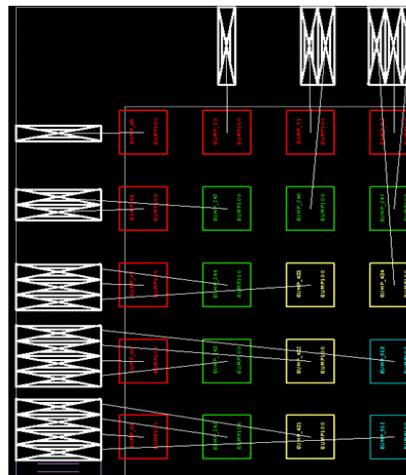
- The core power bumps are arrayed over the core area.

Figure A-8 Placing I/O Signal Nets and I/O Power and Ground Nets



[Figure A-9](#) shows an example of the connectivity flylines between bump nets and I/O drivers.

Figure A-9 Assigning Bump Nets: Connectivity Between Bump Nets and I/O Drivers



Note:

Once a net is assigned between the flip-chip bump and I/O driver, the bump pin is visible in the GUI.

Merging Multiple Flip-Chip Nets Into One Net

You can merge the flip-chip nets that were unqualified by the `assign_flip_chip_nets` command into one net by using the `merge_flip_chip_nets` command.

The `merge_flip_chip_nets` command uses the following syntax:

```
merge_flip_chip_nets
  -from from_nets
  -to to_net
  [-update_routing]
```

The command disconnects all the components, including pins, I/O ports, and terminals, from a specified collection of flat nets and reconnects them to the newly merged net.

Note:

You should use the `-update_routing` option if these nets were partially routed using the `route_flip_chip` command. This updates the owner attribute of the net shapes and vias of the specified nets in the collection.

Assigning Bump Pattern Personality Types

Before you can implement your flip-chip design, you must make sure that flip-chip drivers and bump cells are assigned with the correct bump pattern “personality” type. This allows you to guide the net assignment process.

You use the `set_flip_chip_type` command to assign bump pattern personality types to the flip-chip bump cells, I/O drivers, and specified nets or cells. A “personality type” is a string that is associated with a flip-chip driver or a flip-chip bump cell. You can run this command multiple times; a new personality type setting will override an existing personality type setting.

The `set_flip_chip_types` command uses the following syntax:

```
set_flip_chip_type
  -personality_type type
  net_or_cell_list
```

The following command, for example, selects all the signal bumps and I/O drivers and assigns the personality type "signal."

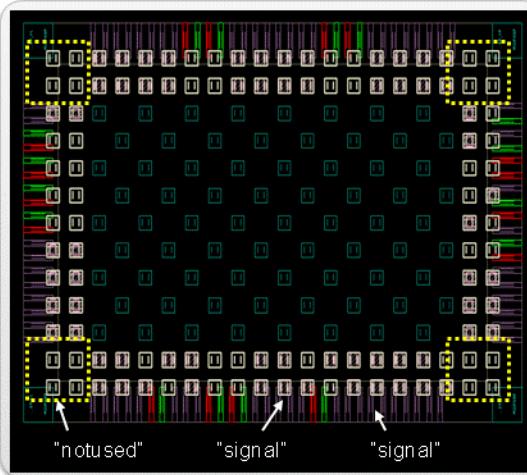
```
set_flip_chip -personality "signal" [get_selection]
```

Another example is the following command, which selects the 4 x 4 bumps at each corner and assign the personality type "notused."

```
set_flip_chip_type -personality_type "notused" [get_selection]
```

[Figure A-10 on page A-13](#) shows a selection of "signal" and "unused" personality types.

Figure A-10 Using Personality Types to Guide Bump Net Assignment

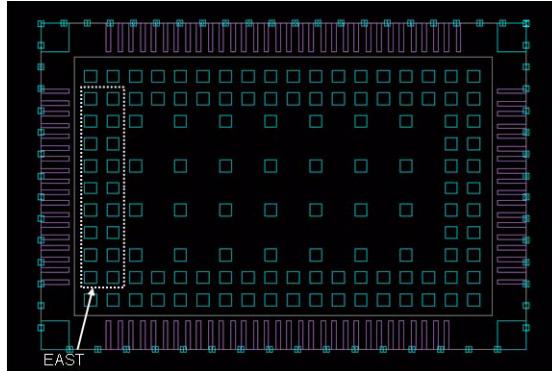


The following command sets the flip-chip bumps to the “east” personality type.

```
set_flip_chip_type -personality_type east [get_selection]
```

[Figure A-11](#) shows a selection of bump cells set to the "east" personality type.

Figure A-11 Selection of Bump Cells for Personality Type Assignment



Using a Package-Driven (Bump-Driven) Bump and Driver Placement Flow

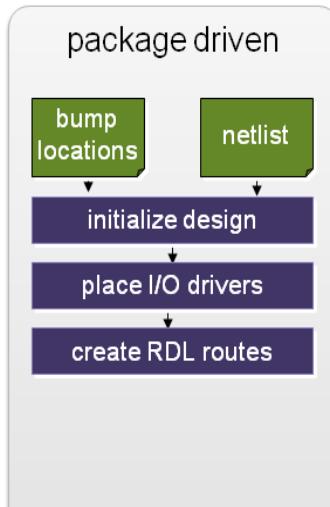
In a package-driven flow, the bump cells are imported and placed at physical locations predefined by the flip-chip package designer.

- The bump cells locations are defined in Design Exchange Format (DEF) or Advanced Input Format files.

- The bump cells instances, I/O driver instances, and the net connectivity between the bump cells and I/O drivers are defined in the Verilog netlist as input or output.
- The I/O drivers are placed as close as possible to their associated bump cells.
- Once the bump cells and I/O drivers are placed and the connectivity is defined, the redistribution layer routes are formed to connect the bumps to the I/O drivers.

[Figure A-12](#) shows the package-driven design flow style.

Figure A-12 Package-Driven Design Flow Style



The following topics are described in this section:

- [Importing and Placing Bump Cells](#)
- [Saving the Flip-Chip Bump Cells](#)
- [Creating a Secondary Grid for Flip-Chip Driver Placement](#)
- [Defining Legal Locations for Placing Flip-Chip Drivers](#)
- [Setting Options for Flip-Chip Driver Placement](#)
- [Placing Flip-Chip I/O Drivers Concurrently With Standard Cells and Hard Macros](#)
- [Reporting Flip-Chip Driver to Bump Results](#)

Importing and Placing Bump Cells

In a package-driven flow, bump cells are imported and placed in predefined physical locations as dictated by the flip-chip package house. The `read_flip_chip_bumps` command reads the bump locations from a text file in the Advanced Input Format file.

You can use the `-physical_lib_cell` option to specify a collection of physical library cells. The collection must contain one cell whose location serves as the reference point from which the flip-chip bump cells are placed in the die area. The default orientation of the bump cells is N (north).

You can use the `-ignore_assign_nets` option if you do not want the bump cells to be connected to the specified nets. The bump cells are placed only at specified locations.

The `read_flip_chip_bumps` command uses the following syntax:

```
read_flip_chip_bumps  
-physical_lib_cell phys_lib_cels  
[-orientation N | W | S | E | FN | FE | FS | FW]  
[-ignore_assign_nets]  
file_name
```

IC Compiler only reads the bump locations from the [NETLIST] section of the AIF file.

Here is an example:

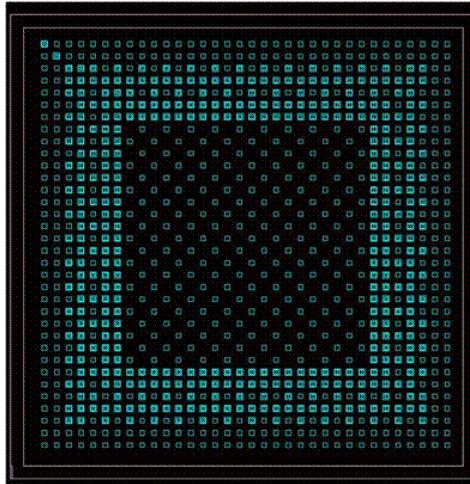
```
read_flip_chip_bumps -physical_lib_cel [get_physical_lib_cells "BUMP"]  
BumpPattern.aif
```

Here is an example AIF file:

```
[NETLIST]  
;NETNAMEDIE_PAD# DIE_PADSTACK_NAME DIE_PAD_X Die_PAD_Y  
GND BUMP_103 BUMP 3795.0 -3105.0  
VD33 BUMP_104 BUMP 3565.0 -3105.0  
TE1_ON_4 BUMP_106 BUMP 3105.0 -3105.0  
TEX_ON_4 BUMP_107 BUMP 2875.0 -3105.0  
DATA_4[19] BUMP_116 BUMP 805.0 -3105.0  
DATA_4[24] BUMP_117 BUMP 575.0 -3105.0
```

[Figure A-13 on page A-16](#) shows the placement locations of the imported bump cells.

Figure A-13 Imported Bump Cells



Saving the Flip-Chip Bump Cells

You can write bump cell information, including the locations and net connections, to a text file in the Advanced Input format (AIF), by using the `write_flip_chip_bumps` command. You must specify the name of the output file.

The `write_flip_chip_bumps` command uses the following syntax:

```
write_flip_chip_bumps  
file_name
```

Creating a Secondary Grid for Flip-Chip Driver Placement

As described earlier, you can use TDF to place flip-chip drivers. IC compiler also supports the automatic placement of flip-chip drivers. To enable automatic placement, you must first create a secondary placement grid. Legal locations of the flip-chip drivers can be restricted to a secondary placement grid.

Typically, the legal locations of the flip-chip drivers are restricted to a secondary placement grid, which is different from the standard placement grid used for standard cell rows and columns.

You use the `set_flip_chip_grid` command to create a secondary placement grid with equally spaced grid points on which to place the flip-chip drivers. This secondary grid is the minimum matrix pitch defined for the actual flip-chip driver placement.

The `set_flip_chip_grid` command uses the following syntax:

```
set_flip_chip_grid
  -grid_origin {llx lly}
  -x_step x
  -y_step y
```

A flip-chip driver must be placed on the secondary placement grid, and all flip-chip driver legal location sites must also reside on this secondary grid.

In the following example, a secondary flip-chip placement grid is created starting at the grid origin of (0, 0) with spacing between the grid of 10 microns in both the X-direction and the Y-direction.

```
set_flip_chip_grid -grid_origin {0 0} -x_step 10 -y_step 10
```

Defining Legal Locations for Placing Flip-Chip Drivers

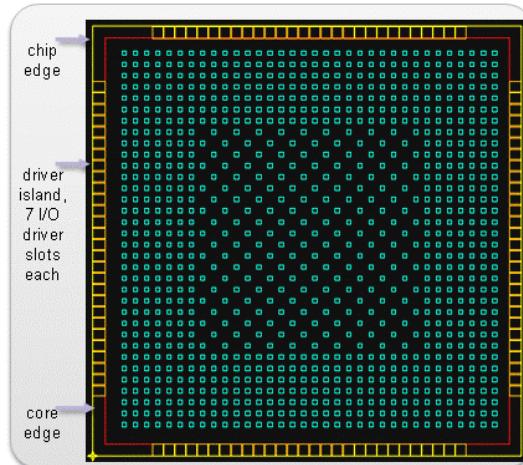
You can define the flip-chip cell sites that form the legal locations for placing the flip-chip drivers in an "island" style and you can define the constraints on the flip-chip driver placement by using the `set_flip_chip_driver_island` command. An island is an array of flip-chip driver locations that are abutted together. You can define an array of islands using a single `set_flip_chip_driver_island` command.

An island can contain a two-dimensional array of flip-chip driver slots. Each slot can hold a unit sized flip-chip driver. Slots within I/O driver islands can be constrained to allow the placement of specific types of drivers.

The `set_flip_chip_driver_island` command uses the following syntax:

```
set_flip_chip_driver_island
  -start_point start_point
  -repeat num_columns num_rows
  -spacing x_spacing y_spacing
  -island_size width height
  -num_driver num_x num_y
  [-filler cell_ref]
  [-personality_type_constraints personality_type max_num ...]
  [-default_orientation N | W | S | E | FN | FE | FS | FW]
  [-orient_by_row]
  [-compaction vertical | horizontal | none]
  [-center_packing]
  [-forced_orientation orientation col_index ...]
  [-reserved_for_cell cell_ref col_index row_index ...]
```

[Figure A-14](#) shows the I/O driver islands and bumps.

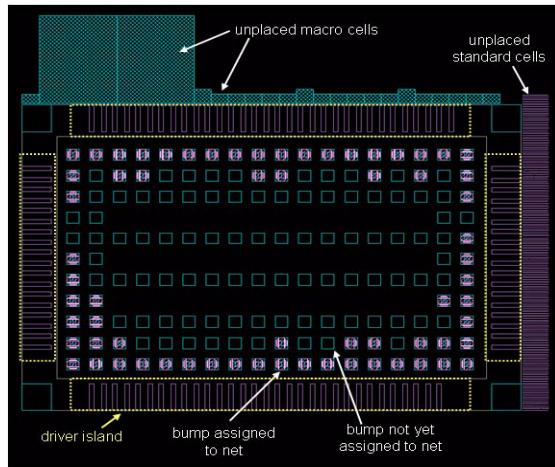
Figure A-14 Creating I/O Driver Islands**I/O driver islands and bumps**

The following example shows a peripheral I/O style where 4 driver islands are needed: one for the left side, one for the right side, one for the top, and one for the bottom. This example creates a flip-chip driver island for the left side.

```
set_flip_chip_driver_island -start_point {0 465} -repeat {1 1} -spacing
{0
0} -island_size {190 1320} -num_driver {1 18}
-personality_type_constraints
{{east 10} {VDD 2} {VSS 2} {VDDPST 2} {VSSPST 2}} -default_orientation E
```

The flip-chip driver island starts at the coordinates (0 465). 18 flip-chip drivers are assigned to the island using different personality types: 10 of east personality, 2 of VDD, 2 of VSS, 2 of VDDPST and 2 of VSSPST. All flip-chip drivers instances will be placed in an E (east) orientation.

Figure A-15 shows a design prior to placement. At this stage in the design flow, the bump patterns exist, nets are assigned between the flip-chip bumps and the I/O drivers, and the driver islands are created.

Figure A-15 Design Prior to Placement

Setting Options for Flip-Chip Driver Placement

Prior to placing the flip-chip drivers and standard cells, you can set various flip-chip driver placement options by using the `set_flip_chip_options` command. You can enable a special optimization algorithm for placing flip-chip drivers based on fixed-bump location constraints by specifying the `-package_driven` option. When multiple width flip-chip drivers are placed, you can specify the `-multiple_width` option.

The `set_flip_chip_options` command uses the following syntax:

```
set_flip_chip_options
[-disable_driver_placement]
[-package_driven]
[-flip_chip_net_weight weight]
[-softmacro_net_weight sm_weight]
[-guardband_width [x y]]
[-multiple_width | height | both]
```

The following example sets a net weight value of 15 on the nets that connect the I/O drivers and the bump cells.

```
set_flip_chip_options -flip_chip_net_weight 15 -package_driven
```

Placing Flip-Chip I/O Drivers Concurrently With Standard Cells and Hard Macros

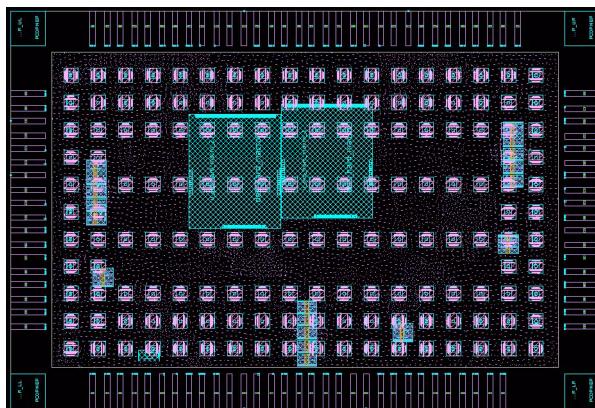
You can place the flip-chip I/O drivers concurrently with hard macros and standard cells in a virtual flat placement mode by using the `create_fp_placement` command. When you run this command, the virtual flat placement engine calls the driver placement under the flip-chip mode to place the flip-chip I/O drivers into defined legal sites and snap them to a secondary grid. To set the flip-chip mode, enter:

```
set_fp_placement_strategy -name flip_chip -value 1
```

The objective of flip-chip driver placement is to enhance single-layer routability, while optimizing total wire length. The even distribution of I/O power drivers at the desired signal-to-power ratio can further be assured by defining and repeating cell sites by using the `-reserve_for_cell` option in the `set_flip_chip_driver_island` command.

[Figure A-16](#) shows the simultaneous placement results of the flip-chip drivers, hard macros, and standard cells.

Figure A-16 Simultaneous Placement of Flip-Chip Drivers, Hard Macros, and Standard Cells



Reporting Flip-Chip Driver to Bump Results

You can issue a report in a table format of the flip-chip driver to bump matching results by using the `report_flip_chip_driver_bump` command.

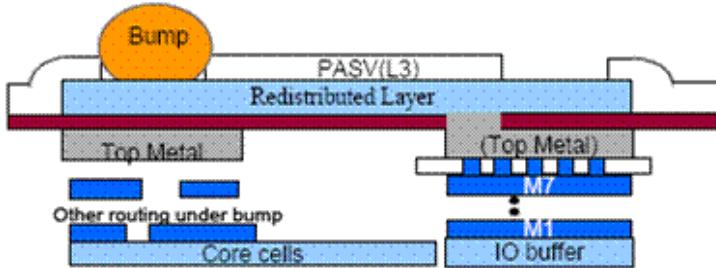
Each line in the table represents a flip-chip net connection. The first column specifies the bump cell; the second column lists the name of the net that connects the bump cell and the flip-chip I/O driver; and the third column specifies the driver cell.

Performing Automatic Bump Net Routing

Before you can perform regular signal detailed routing, the bump nets must be routed. Bump net routing, also referred to as redistribution layer routing, redistributes the wire-bonding pads to the bump pads without changing the placement of the I/O pads. Typically, bump nets are routed on the top metal layer of the die.

[Figure A-17](#) shows a cross-section of redistribution layer routing.

Figure A-17 Cross-Section showing Redistribution Layer Routing



Note:

For peripheral I/O flip-chip design styles, single-layer routing on the top metal layer is preferred. In addition, single-layer routing on the top metal layer minus one level (dual-layer routing), is also allowed when necessary.

The following topics are described in this section.

- [Using the Flip-Chip Routing Commands](#)
- [Routing the Flip-Chip Nets](#)

Using the Flip-Chip Routing Commands

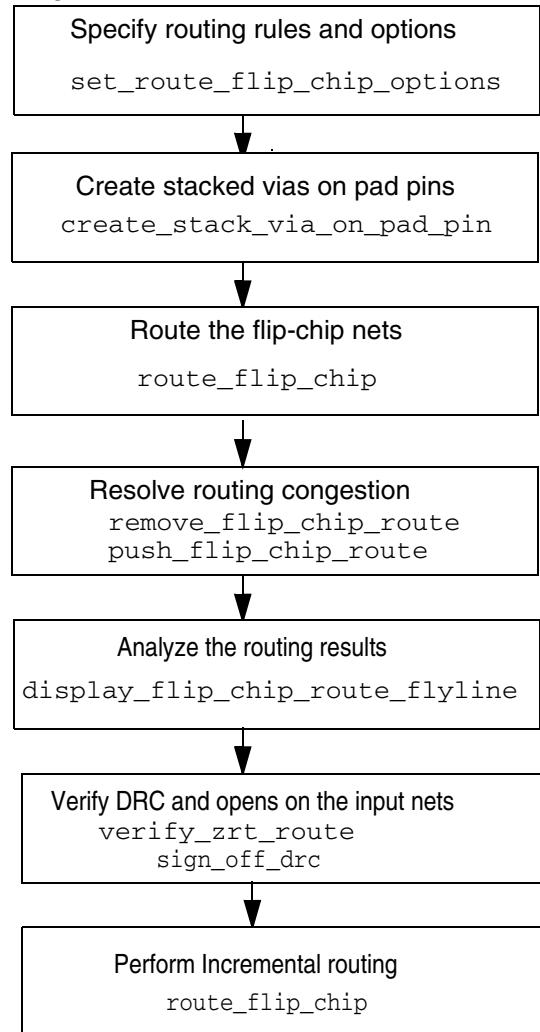
This section describes the usage of the following flip-chip routing commands.

- `set_route_flip_chip_options`
- `create_stack_via_on_pad_pin`
- `route_flip_chip`
- `remove_flip_chip_route`
- `push_flip_chip_route`
- `display_flip_chip_route_flyline`

- `verify_zrt_route`
- `sign_off_drc`

You can use flip-chip routing commands to complete the flip-chip routing flow, shown in [Figure A-18](#).

Figure A-18 Flip-Chip Routing Flow



Specifying Routing Rules and Options

Use the `set_route_flip_chip_options` command to set options for flip-chip routing. The options are stored in the design database for use by subsequent flip-chip routing steps.

The `set_route_flip_chip_options` command uses the following syntax:

```
set_route_flip_chip_options
[-number_of_loops 1 | 2 | 3]
[-route_with_soft_macros true | false]
[-touch_pins_at_center true | false]
[-search_effort easy | normal | hard]
[-layer_spacing {layer_spacing_pairs}]
[-layer_width {layer_width_pairs}]
[-rule_name name]
[-nets {collection_of_nets} | -nets_in_file net_file]
[-routing_layer layer]
[-output_unrouted_nets open_net_file]
[-design_style area_IO | peripheral_without_corner_driver |
peripheral_with_corner_driver]
```

Creating Stack Vias on Pad Pins

Use the `create_stack_via_on_pad_pin` command to create stack vias on the flip-chip I/O drivers from the pad pin to the top metal layer for flip-chip routing. Stack vias must be created before you route the flip-chip nets. (For more information, see “[Adding Stack Vias to Driver Pins to Enable Single Layer Routing](#)” on page A-27.)

The `create_stack_via_on_pad_pin` command uses the following syntax:

```
create_stack_via_on_pad_pin
[-from_metal mX | MX | tech_layer_number]
[-to_metal me | MX | tech_layer_number]
[-remove_existing_stack_via]
[-route_type User | Detail]
```

Performing Redistribution Layer Routing

Use the `route_flip_chip` command to perform flip-chip redistribution layer routing on the flip-chip nets. (For more information, See “[Routing the Flip-Chip Nets](#)” on page A-27.)

The `route_flip_chip` command uses the following syntax:

```
route_flip_chip
[-nets {collection_of_nets} | -nets_in_file net_file]
[-45_degree]
[-90_degree]
[-route_by_input_net_order]
[-compact_routing]
```

Resolving the Routing Congestion

Use the `remove_flip_chip_route` command to resolve the routing congestion by removing specified flip-chip wires, paths, and contacts.

You can resolve routing congestion from routing by

- Removing the nets that create the bottleneck net

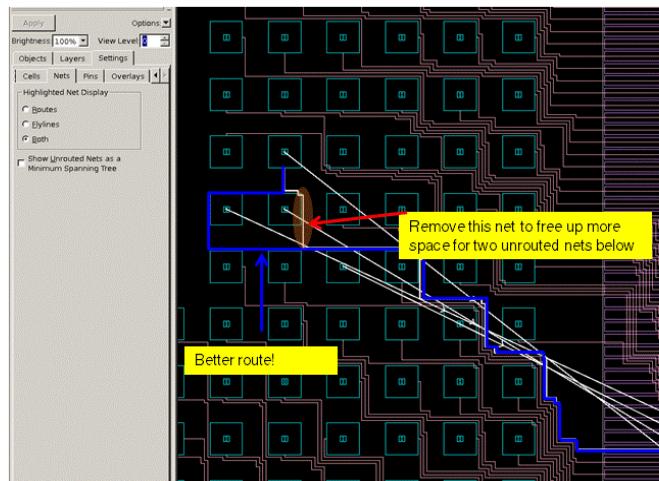
- Pushing nets away from the congested area to create more routing resources

You can resolve routing congestion from placement by

- Rearranging the placement of the I/O pads or the bump cells or both
- Redoing the net assignment

[Figure A-19](#) shows an example of removing a bottleneck net.

Figure A-19 Removing Nets that Create the Bottleneck



The `remove_flip_chip_route` command uses the following syntax:

```
remove_flip_chip_route
[-nets {collection_of_nets} | -nets_in_file net_file]
[-width width]
[-contact]
```

Use the `push_flip_chip_route` command to push the routed flip-chip nets that block the path of unrouted nets in a specified direction or away from the specified nets.

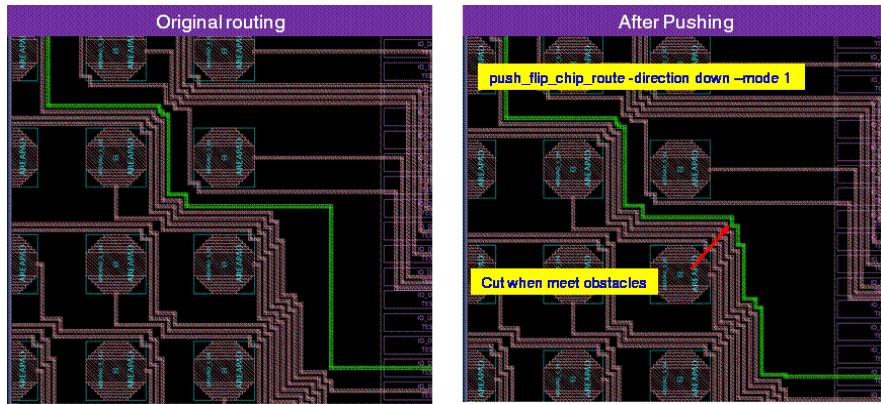
The `push_flip_chip_route` command uses the following syntax:

```
push_flip_chip_route
[-nets {collection_of_nets} | -nets_in_file net_file]
[-layer mX | MX | tech_layer_number]
[-direction [up | down | left | right]
[-mode 1 | 2 | 3]
[-sweep_range sweep_range]
[-all]
```

This command supports the following three modes of pushing the routed flip-chip nets:

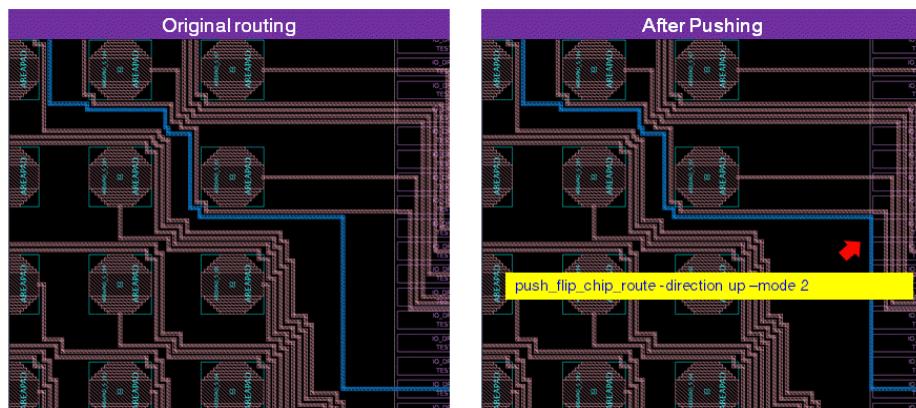
- Mode 1 – Pushes the routed nets toward a specified direction (up, down, left, or right) to free up more space for the unrouted nets. This is the default. [Figure A-20 on page A-25](#) shows an example of push mode 1.

Figure A-20 Push Mode 1

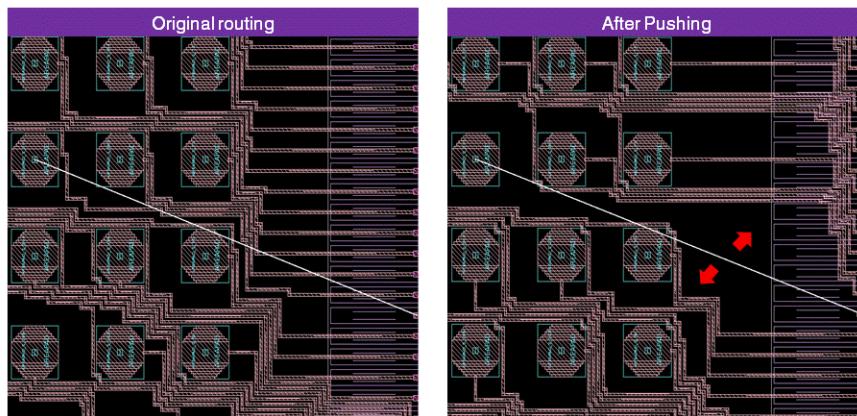


- Mode 2 – Pushes the target net along the edge of the flip-chip I/O drivers. [Figure A-21 on page A-25](#) shows an example of push mode 2.

Figure A-21 Push Mode 2



- Mode 3 – Pushes neighboring wires away from the target net. [Figure A-22 on page A-26](#) shows an example of push mode 3.

Figure A-22 Push Mode 3

Optimizing the Routing Pattern

Use the `optimize_flip_chip_route` command to improve and enhance the flip-chip routing patterns by running L-shape and Z-shape optimizations on the flip-chip nets. The command routes the flip-chip nets on a single metal layer.

The `optimize_flip_chip_route` command uses the following syntax:

```
optimize_flip_chip_route
[-nets collection_of_nets]
[-layer m1 to m15 | technology_layer_number]
[-change_route_type user_enter | signal_route]
[-nets_in_file nets_file]
[-convert_path_to_polygon]
```

Analyzing the Routing Results

Use the `display_flip_chip_route_flylines` command to display the flylines of the specified flip-chip nets in the layout window of the IC Compiler GUI. You can view unrouted nets with flylines and identify any nets that are creating bottlenecks.

The `display_flip_chip_route_flylines` command uses the following syntax:

```
display_flip_chip_route_flylines
[-nets {collection_of_nets} | -nets_in_file nets_file]
[-open_nets [-output_open_nets open_net_file]]
```

Verifying Design Rule Checking and Open Nets

Use the `verify_zrt_route` command to check and report design rule checking (DRC) violations and open nets on the input bump nets.

The `verify_zrt_route` command uses the following syntax:

```
verify_zrt_route
[-nets collection_of_nets]
[-open_net true | false]
[-report_all_open_nets true | false]
[-drc true | false]
[-antenna true | false]
[-voltage_area true | false]
[-check_from_user_shapes true | false]
```

Use the `signoff_drc` command to detect 65 nm and below process design rule-checking violations with foundry runset.

Routing the Flip-Chip Nets

The flip-chip nets are routed by using the `route_flip_chip` command. Before running this command, however, it may be necessary to perform the following setup procedures.

- Specifying the Names of the Bump Nets for Routing

Use the `write_flip_chip_nets` command to write all the bump-to-driver net names into a text file. You can edit this file to specify a subset of the bump nets to be routed.

For example,

```
write_bump_to_pad_nets -file_name bump_net_list
```

- Defining Special Routing Rules

Use the `set_route_flip_chip_options` command to set various flip-chip routing options, and use the `define_routing_rule` command to save them as nondefault routing rules.

For example,

```
set_route_flip_chip_options -rule_name test -layer_spacing {m8 2.0}
-layer_width {m8 12}
```

- Adding Stack Vias to Driver Pins to Enable Single Layer Routing

Use the `create_stack_via_on_pad_pin` command to create stack vias on the flip-chip I/O drivers from the pad pin to the top metal layer for flip-chip routing. The flip-chip router will then route from the stack via on the redistribution layer to the bumps on the same layer.

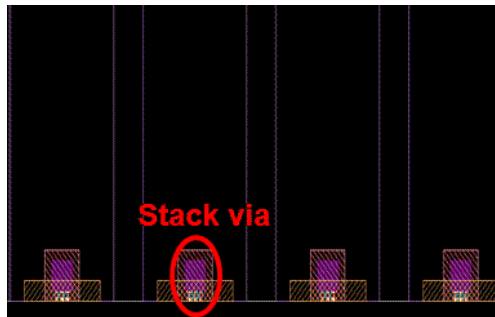
The stack vias are created based on the input nets specified in the `set_route_flip_chip_options` command. The size of the stack vias is determined by the predefined nondefault routing rule and the shape of the pad pin.

For example,

```
create_stack_via_on_pad_pin -pad_layer 36 -top_layer 38
```

A stacked via pad is shown in [Figure A-23](#).

Figure A-23 Stack Via Pad



- Perform flip-chip redistribution layer routing on the flip-chip nets

Use the `route_flip_chip` command to perform redistribution layer routing on the flip-chip nets. The command routes the flip-chip nets on a single metal layer.

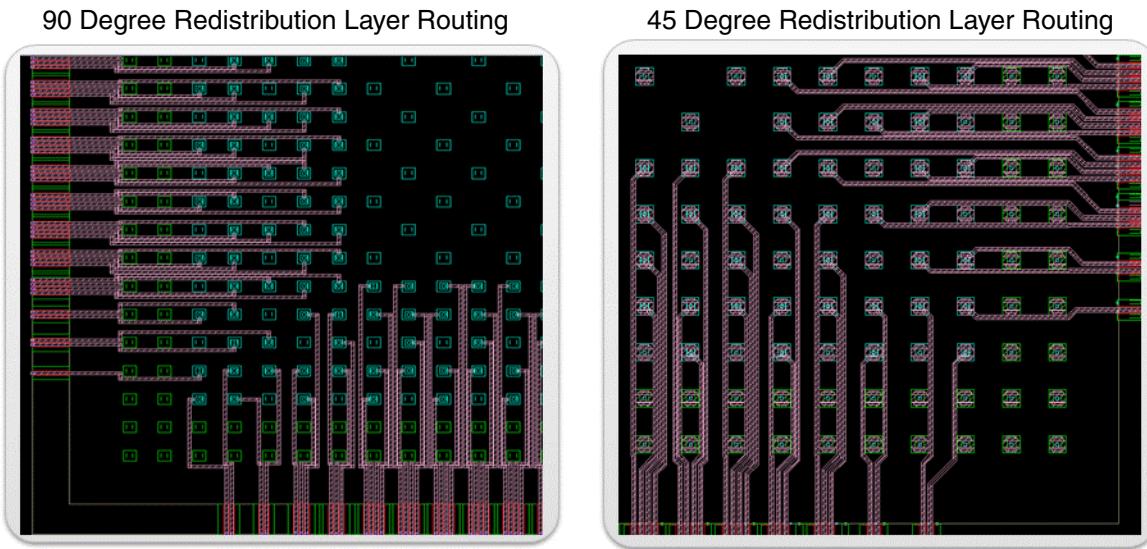
IC Compiler performs 45 degree routing and 90 degree Manhattan routing for flip-chips. By default, 45 degree routing is on.

The following example shows how to set 45 degree redistribution layer routing.

```
route_flip_chip -net_in_file bumpnet -route_top_layer 1  
-route_45_degree 1
```

[Figure A-24](#) shows an example of 45 degree and 90 degree redistribution layer routing.

Figure A-24 45 Degree and 90 Degree Redistribution Layer Routing



Using Flip-Chip Structures in Cover Macros

IC Compiler supports flip-chip bump and cover cells in the logical netlist. You can describe the logical connectivity between the flip-chip bump pads and cover cells and the I/O drivers in the Verilog netlist.

You can change the flip-chip bump and cover cells from physical-only by cell type to become part of the logical netlist by setting the following variable to a value of FALSE.

```
set disable_bump_cover_as_physical_only TRUE
```

Summary of the Flip-Chip Commands

[Table A-1](#) provides a summary of the flip-chip commands. For more information, see the appropriate man pages.

Table A-1 Summary of Flip-Chip Commands

Command	Description
assign_flip_chip_nets	Creates or reconnects nets between flip-chip I/O drivers to bumps.
create_stack_via_on_pad_pin	Creates stack vias for the I/O drivers from the pad layer to the top metal layer for flip-chip routing.
display_flip_chip_route_flylines	Displays the flylines of the specified flip-chip nets in the layout window of the IC Compiler GUI. You can view unrouted nets with flylines and identify any nets that are creating bottlenecks.
expand_flip_chip_cell_locations	Proportionally expands or shrinks the relative locations of the selected flip-chip bump or driver cells.
merge_flip_chip_nets	Disconnects all the components, including pins, I/O ports, and terminals, from a specified collection of flat nets and reconnects them to the newly merged net.

Table A-1 Summary of Flip-Chip Commands

Command	Description
optimize_flip_chip_route	Improves and enhances the flip-chip routing patterns by running L-shape and Z-shape optimizations on the flip-chip nets. The command routes the flip-chip nets on a single metal layer.
place_flip_chip_array	Creates the specified number of flip-chip bump cells and places them in a two-dimensional array pattern.
place_flip_chip_ring	Creates the specified number of flip-chip bump cells and places them in a ring configuration.
push_flip_chip_route	Pushes the routed flip-chip nets that block the path of unrouted nets in a specified direction or away from the specified nets.
read_flip_chip_bumps	Reads the flip-chip bump locations from a Design Exchange Format (DEF) file or from a Advanced Input Format file.
remove_flip_chip_route	Helps resolve the routing congestion by removing specified flip-chip wires, paths, and contacts.
report_flip_chip_driver_bump	Reports the flip-chip of the driver-to-bump matching results.
report_flip_chip_type	Reports the personality types of specified nets, bumps, or I/O drivers.
route_flip_chip	Performs flip-chip redistribution layer routing to connect the flip-chip I/O driver and the bump cell.
set_flip_chip_cell_site	Modifies flip-chip driver cell site properties to set different flip-chip driver legal location constraints.
set_flip_chip_driver_array	Defines the legal locations for flip-chip driver cells to be placed in an array configuration.
set_flip_chip_driver_island	Defines the flip-chip cell sites that form the legal locations for the flip-chip drivers to be placed in an island style.

Table A-1 Summary of Flip-Chip Commands

Command	Description
set_flip_chip_driver_ring	Defines the legal locations for flip-chip driver cells to be placed in a ring configuration
set_flip_chip_grid	Creates equally-spaced grid points for flip-chip driver placement.
set_flip_chip_options	Sets general flip-chip driver placement options during virtual flat placement.
set_flip_chip_type	Assigns personality types to specified nets, flip-chip bumps, or flip-chip drivers.
set_route_flip_chip_options	Defines the flip-chip routing options.
update_flip_chip_pin_locations	Updates the flip-chip bump I/O pin locations for timing analysis.
write_flip_chip_nets	Writes all bump to pad net names into a file.

Index

A

acceptable overflow criteria 3-14
add_row command 2-15
advanced via rules, setting 7-35
align_objects command 5-38
attributes
 mtcmos_resistance 7-41, 7-43, 7-45

B

black boxes
 create FRAM view 4-2
 creating CEL views for 4-3
 creating quick timing models for 4-7
 estimating sizes of 4-5
 flattening existing instances 4-7
 managing the properties for 4-7
 module types
 Data flow 4-3
 Empty 4-2
 Feedthrough 4-3
 Missing 4-2
 Tie-off 4-3
 read verilog netlist 4-2
 sizing by gate equivalence 4-5
block level designs, performing simultaneous placement and pin assignment 5-39

C

CEL view, working with soft macros 13-9
CEL views
 creating for black boxes 4-4
cell rows
 adding 2-15
 creating for rectilinear floorplan 2-15
 removing 2-16
cells (macro, standard), moving to new location 5-36
channel resizing, using parameters to control 6-20
check_timing command 2-15
clock
 latency 9-5, 12-15
clock planning
 performing plan group-aware clock tree synthesis 9-8
 using multivoltage designs 9-7
clock skew analysis 9-4
coarse-grain, fine-grain switching strategy 7-40
command
 explore_header_footer 7-41
commands
 add_header_footer_cell_array 7-41
 add_row 2-15
 adjust_fp_io_placement 2-7
 analyze_fp_rail 7-97

analyze_fp_routing 10-3
check_timing 2-15
commit_fp_rail 7-69, 7-93
connect_net 7-14, 7-15
connect_power_switch 7-48
connect_tie_cells 7-16
connect_virtual_pg_net 7-44
creat_fp_virtual_pad 7-100
create_connview 7-104
create_fp_placement 5-28, 5-34
create_fp_virtual_pad 7-43, 7-45
create_placement_blockage 5-26
create_plan_groups 6-3
create_power_straps 7-24, 7-93
create_preroute_vias 7-36
create_qtm_constraint_arc 4-11
create_qtm_drive_type 4-10
create_qtm_load_type 4-10
create_qtm_model 4-10
create_qtm_port 4-11
create_rectangular_rings 7-18
create_rectilinear_rings 7-22
create_route_guide 8-2
create_voltage_area 5-29
cut_row 2-16
define_routing_rule 8-6
derive_pg_connection 7-6, 7-11, 7-41, 7-44, 7-45
estimate_fp_area 3-1, 3-7, 3-14, 3-25
estimate_fp_black_boxes 4-5
explore_header_footer 7-41, 7-42
extract_fp_relative_location 5-15
flatten_fp_black_boxes 4-7
get_fp_wirelength 5-51
get_route_guides 8-4
get_voltage_area 5-33
import_fp_black_box 4-4
initialize_floorplan 2-8
legalize_fp_placement 5-36
optimize_fp_timing 10-2
optimize_header_footer 7-45, 7-46
pack_fp_macro_in_area 5-37
place_io_pads 2-5
preroute_standard_cells 7-29
read_io_constraints 2-2
read_saif 7-109
read_sdc 2-15
read_verilog 4-2
remove_fp_rail_voltage_area_constraints 7-66
remove_fp_relative_location 5-16
remove_placement 5-36
remove_plan_groups 6-4
remove_preferred_routing_direction 8-5
remove_route_by_type 8-8
remove_route_guides 8-4
replace_header_footer 7-43, 7-44
report_fp_placement 5-51
report_fp_placement_strategy 5-25
report_fp_rail_strategy 7-124
report_fp_relative_location 5-15
report_preferred_direction 8-5
report_qtm_model 4-11
report_route_options 8-2
report_timing 4-8
route_fp_proto 8-9
set_attribute 7-41, 7-43, 7-53, 7-59, 7-94
set_fp_base_gate 4-5
set_fp_black_boxes_estimated 4-7
set_fp_black_boxes_unestimated 4-7
set_fp_block_ring_constraints 7-83
set_fp_macro_array 5-5
set_fp_pin_constraints 11-18
set_fp_rail_region_constraints 7-85
set_fp_rail_voltage_area_constraints 7-51, 7-58, 7-62, 7-79, 7-96
set_fp_relative_location 5-14
set_fp_voltage_area_constraints 7-68
set_ignored_layers 8-5
set_keepout_margin 3-11, 5-26
set_net_aggressors 8-8
set_net_routing_rule 8-7
set_preferred_routing_direction 8-4
set_preoute_advanced_via_rule 7-21

set_preroute_advanced_via_rule 7-35
 set_preroute_drc_strategy 7-32
 set_preroute_special_rules 7-34
 set_qtm_global_parameter 4-11
 set_qtm_port_drive 4-11
 set_qtm_technology 4-10
 set_route_options 8-2
 set_route_type 8-7
 set_switching_activity 7-108
 shape_fp_block 6-15
 shape_fp_blocks multiple instantiated modules
 shaping plan groups in (shape_fp_blocks) 5-45
 synthesize_fp_rail 7-67, 7-73
 update_voltage_area 5-33
 write_floorplan 2-21
 write_io_constraints 2-14
 write_qtm_model 4-11
 CONN views, creating for power and ground 7-103
 constraints
 power network synthesis
 specifying global 7-79
 reading 2-15
 core area, defining 2-8
 core ring and strap constraints, setting for power planning 7-80
 core rows, configuring 2-8
 core utilization factor, defined 2-8
 Create Voltage Area dialog box, setting the options 5-29
 create_route_guide command 8-2
 creating
 pin guides 11-33
 crosstalk
 preventing nonanalytical 11-2
 crosstalk prevention, net aggressors 8-8
 cut_row command 2-16

D
 daisy-chain mode 7-49, 7-50
 DEFAULT_VA 7-68
 define_routing_rule command 8-6
 design rule checking, changing internal rules for power and ground commands 7-32
 detecting during in-place optimization 10-6
 distribute_objects 5-39
 distributing objects 5-39

E
 electromigration violations, displaying 7-113
 Estimate Area GUI
 floorplan control options 3-8
 using 3-8
 using power network control options 3-14
 using search control options 3-14
 estimate_area options
 high_utilization_bound 3-19
 pre_route_post_pns 3-24
 run_pns_script 3-24
 save_as_cell 3-24

F
 feedthrough ports, removing ports and nets 11-41
 fine-grain, coarse-grain switching strategy 7-40
 floorplan control options
 increase spacing in the core area 3-13
 maintain I/O pad alignment 3-12
 recognizing blockages 3-10
 select sizing type 3-9
 specify core area boundaries 3-9
 floorplan file
 writing 2-20
 floorplan physical constraints, writing for topographical technology 2-22

floorplan, saving 2-20

G

get_route_guides command 8-4
get_voltage_area command 5-33
global constraints, setting for power planning 7-79

H

hard macro placement
 measuring QoR 5-2
 specifying constraints 5-3
hard macros
 automatic padding 5-26, 5-27
 controlling the placement of 5-33
 creating a user-defined array 5-3
 creating macro blockages formacro
 blockages, creating for hard macros 5-25
 manually adjusting 5-38
 packing into an area 5-37
 placing (and standard cells) 5-28
 setting a user-defined padding distance
 (keepout margin) 5-26
hierarchical timing views (HTV) 11-2
high-fanout mode 7-49, 7-50
high-fanout synthesis
 threshold 10-2

I

I/O constraints
 writing 2-14
I/O locations, adjusting 2-6
I/O pads
 placing 2-7
 working with multiheight 2-5
I/O style, specifying 2-5
icc_shell
 invoking PrimeRail within IC Compiler 7-126

in-place optimization

 detecting multivoltage settings 10-6
 pin cutting 10-6
in-place optimization, running 10-2

K

keepout margins 5-26

L

layer constraints
 specifying for power planning 7-82
level shifters 9-7
limited soft macro flattening (LSMF) 11-2

M

macro cell
 placing relative to anchor object 5-14
Macro Constraints dialog box 5-5
MinChip
 using multivoltage designs 3-25
MinChip technology 3-1
 estimate_area 3-1
 performing steps inside 3-2
preparing the design
 blockages over hard macros 3-7
 edge blockages 3-7
 filler cells 3-8
 floorplan 3-4
 hard macro packing 3-6
 placement blockages 3-7
 power network synthesis scripts 3-4
 preroute standard cell scripts 3-5
 relative hard macro placement 3-5
 sliver size 3-6
 using 3-1
multiheight I/Os 2-5
multiple instantiated modules
 placement of 5-43

multiple operating voltages 10-6
 multiple voltage areas, creating a floorplan with 5-35
 multithreshold CMOS cells
 placing in a column style 7-56
 placing in a ring pattern 7-57
 placing in an array pattern 7-54
 multithreshold CMOS designs
 specifying placement styles 7-54
 multithreshold_CMOS designs
 creating virtual power and ground nets and connecting power power-switching cells 7-44
 multithreshold-CMOS
 connect_virtual_pg_net 7-44
 multithreshold-CMOS designs
 connecting switch pins 7-48
 header and footer exploration 7-41
 inserting header and footer cells 7-42
 optimizing power switching cells 7-45
 performing power network analysis on power-switching cells 7-47
 replacing existing header or footer cells 7-43
 simulating the IR drop across switch cells 7-59
 multivoltage constraints, defining 7-62
 multivoltage designs
 nested voltage areas 5-31
 performing power network synthesis on 7-50
 using in clock planning 9-7
 using in MinChip 3-25

N

nested voltage areas, multivoltage designs 5-31
 net aggressors 8-8
 net layer constraints, specifying 8-5
 nondefault routing rules, setting 8-6
 non-DEFAULT_VA 7-68

O
 objects
 placing and shaping in design core 6-14
 optimizing pins for block level designs 5-39

P
 packing hard macros 5-37
 pgExportICCDData 7-135
 physical constraints (defining)
 core area and shape 2-23
 macro location and orientation 2-26
 placement keepout 2-27
 port locations 2-25
 physical hierarchy
 commit 1-10, 13-1
 uncommit 13-8, 13-9
 physical objects, connected to power and ground 13-9
 pin assignment
 analyzing quality of 1-10, 11-42
 connectivity driven 11-10
 constraints 11-9
 corner calculation 11-4
 results, analyzing 11-1
 pin constraints
 assign 11-2
 assigning 11-2
 exclude pins on macro 11-5
 pin cutting flow, detecting buses 11-20
 pin cutting, in-place optimization 10-6
 pin guides
 creating 11-33
 pin overlaps, removing 11-16
 place_io_pads 2-5
 placement
 measuring QoR 5-2
 specifying hard macro placement constraints 5-3
 placement constraints
 congestion driven 5-34

effort level 5-34
ignore scan chain connectivity 5-36
incremental placement 5-35
legalizing the placement 5-36
maximum fanout 5-35
optimize pins 5-36
setting on macro cells 5-9
specifying CPUs 5-36
timing driven 5-34
placement options, hierarchical gravity 5-34
plan group-aware routing, detecting buses 11-20
plan groups
 creating 6-2
 defining the shape of 6-3
 exclusive 6-2
 removing 6-4
 uncommit hierarchy 13-1
poCalculateResistivity 7-134
power and ground network, creating CONN views 7-103
power and ground rings, adding 7-18
power and ground straps
 aligning with area I/O cells 7-92
 aligning with standard cell rails 7-91
 aligning with top level pins 7-89
 placing inside standard cell rows 7-91
power and ground straps, adding 7-24
power budget, computing 7-73, 7-97
power budgets, allocating to voltage areas 7-67
power network analysis
 performing 7-97
power network synthesis
 benefits of 7-70
 why use? 7-70
power network synthesis constraints, defining multivoltage 7-62
power network synthesis, supporting UPF mode 7-96
power planning
 setting global constraints 7-79
power planning constraints
 specifying for block ring 7-83
 specifying for multiple core or sandwich core rings 7-81
 specifying for power pad synthesis 7-84
 specifying layer 7-82
 specifying power ring and strap 7-80
power planning estimation 7-69
power-switching cells 7-41
pre-budget timing analysis 12-3
preferred routing direction
 reporting 8-5
 setting 8-4
preoute advanced rules, setting 7-35
preoute DRC options, setting 7-32
preroute rules, defining set of special 7-34
preroute standard cells 7-29
preroute vias, creating 7-36
propagate preroutes 13-1
prototype global routing, running 8-9
pushed up
 physical objects 13-9
pushed up, physical objects 13-9

Q

quick timing model for black boxes
 command summary 12-10
 prerequisites for running 4-9
 using tcl commands to create 4-10

R

read_io_constraints command 2-2
read_sdc command 2-15
real clock latencies, creating budgets for 12-16
rectangular rings, creating 7-18
rectilinear rings, creating 7-22
rectilinear-shaped blocks, creating

- 2-10
 - region constraints, specifying for power network synthesis 7-85
 - relative location constraints
 - extracting 5-15
 - removing 5-16
 - relative placement
 - in virtual flat placement 5-40
 - remove_preferred_routing_direction command 8-5
 - remove_route_by_type command 8-8
 - remove_route_guides command 8-4
 - report_preferred_direction command 8-5
 - report_route_options command 8-2
 - reporting
 - routing options 8-2
 - rings
 - adding power and ground 7-18
 - route guide
 - creating 8-2
 - finding 8-4
 - removing 8-4
 - routing direction, setting preferred 8-4
 - routing options, setting 8-2
 - routing rules (nondefault), setting 8-6
 - routing type
 - removing 8-8
 - routing type, setting 8-7
-
- ## S
- SDC
 - reading 2-15
 - search control options
 - acceptable overflow 3-14
 - auto routability 3-14
 - set_fp_placement_strategy
 - auto_grouping 5-19
 - congestion_effort 5-24
 - fix_macros 5-22
 - IO_net_weight 5-23
 - legalizer_effort 5-24
 - macro_orientation 5-18
 - macro_setup_only 5-18
 - macros_on_edge 5-20
 - plan_group_interface_net_weight 5-24
 - sliver_size 5-21
 - snap_macros_to_user_grid 5-22
 - spread_spare_cells 5-24
 - virtual_IPO 5-25
 - voltage_area_interface_net_weight 5-23
 - voltage_area_net_weight_LS_only 5-23
 - set_ignored_layers command 8-5
 - set_net_aggressors command 8-8
 - set_net_routing_rule command 8-7
 - set_preferred_routing_direction command 8-4
 - set_route_options command 8-2
 - set_route_type command 8-7
 - setting routing types 8-7
 - shaping objects 6-15
 - shutdown voltage area, synthesizing the power network 7-51
 - sliver_size option, using for estimate_area 3-6
 - sliver_size parameter 3-11
 - soft macros
 - discarded pins 11-10
 - exclude pins 11-5
 - no pins created in edge 11-3
 - properties set in CEL view 13-9
 - uncommit physical hierarchy 13-1
 - special preroute rules, defining 7-34
 - standard cells, prerouting 7-29
 - straps
 - adding power and ground 7-24
 - extending to boundary or target 7-26
 - setting pitch within groups 7-26
 - setting step distance 7-25
 - switching activity, annotating 7-108
 - Synopsys Design Constraints (SDC) 2-15

T

Tcl I/O constraints 2-2
tie cells 7-16
tied-off pins, connecting to power and ground pins 7-10
timing budgeting
 creating real clock latencies 12-16
 performing in design planning 12-1
 pre-budget timing analysis 12-3
prerequisites 12-2
running 12-6
timing constraints
 loading 2-15
 removing 2-15
 setting 2-15
track assignment 8-10

U

Unified Power Format (UPF)
 defining power-down voltage area 7-51, 7-52, 7-63
unified power format, supporting in power network synthesis 7-95
update_voltage_area command 5-33

V

via rules, setting preroute advanced 7-35
vias, creating preroute 7-36
virtual flat placement strategy, using constraints to control 5-16
virtual pads, adding 7-100
voltage areas
 allocating power budgets 7-67
 DEFAULT_VA 7-68
 generating core rings and connecting power straps 7-68
 non-DEFAULT_VA 7-68
 planning location and shape of 5-28
 removing constraints 7-66
 removing commands
 remove_voltage_area 5-33
 reporting constraints 7-66
 supporting physical boundary scenarios 5-32
 synthesizing the 7-67
 updating 5-33
voltage drop distribution, checking for 7-111
voltage drop violations, displaying 7-112, 7-114

W

write_floorplan 2-20
write_io_constraints 2-14